Natural Language Processing and Large Language Models

Corso di Laurea Magistrale in Ingegneria Informatica

Lesson 15

Encoder-Decoder Transformers

Nicola Capuano and Antonio Greco

DIEM – University of Salerno

Outline

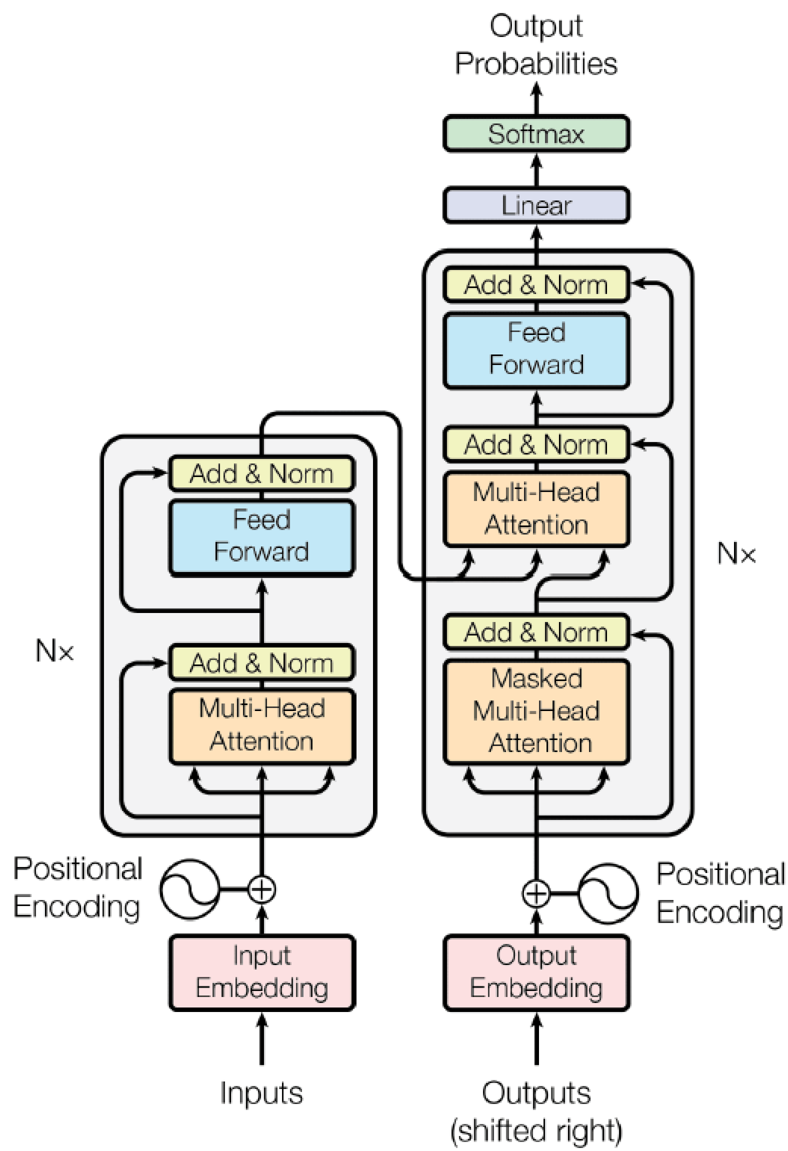
- Encoder-decoder transformer
- T5
- Practice on translation

and summarization

Encoder-decoder transformer

Encoder-decoder transformer

- Encoder-Decoder Transformers are a class of neural networks designed for sequence-to-sequence (seq2seq) tasks.



T5

T5

- **T5** (Text-to-Text Transfer Transformer) is a language model based on an encoder-decoder transformer developed by Google Research.
- T5 comes in multiple sizes to suit different resource constraints:

T5 Version	Encoder Blocks	Attention Heads	Decoder Blocks	Embedding Dimensionality
T5-Small	6	8	6	512
T5-Base	12	12	12	768
T5-Large	24	16	24	1024
T5-XL	24	32	24	2048
T5-XXL	24	64	24	4096

T5 input encoding

- T5 uses a **SentencePiece** tokenizer with a custom vocabulary for its input encoding.
- **Subword Units**: T5 employs a subword-based tokenizer using the SentencePiece library. Subword tokenization ensures a balance between character-level and word-level tokenization, effectively handling rare words and unseen combinations.
- **Unigram Language Model**: The SentencePiece tokenizer in T5 is trained using a unigram language model, which selects subwords to maximize the likelihood of the training data.

T5 input encoding

- T5 uses a fixed vocabulary of 32,000 tokens. The vocabulary includes subwords, whole words, and special tokens. This compact vocabulary size strikes a balance between computational efficiency and representation capacity.

- T5 introduces several special tokens in the vocabulary to guide the model for various tasks:
  - <pad>: Padding token for aligning sequences in a batch.
  - <unk>: Unknown token for handling out-of-vocabulary (OOV) cases.
  - <eos>: End-of-sequence token, marking the conclusion of an input or output sequence.
  - <sep> and task-specific prefixes: Used to define input task types (e.g., "translate English to German:" or "summarize:").

T5 pre-training

- For pre-training, T5 uses a denoising autoencoder objective called **span-corruption**.
- This objective involves masking spans of text (not just individual tokens) in the input sequence and training the model to predict those spans.
- **Input Corruption**: Random spans of text in the input are replaced with a special <extra_id_X> token (e.g., <extra_id_0>, <extra_id_1>).
    - Original Input: "The quick brown fox jumps over the lazy dog."
    - Corrupted Input: "The quick <extra_id_0> jumps <extra_id_1> dog."
- **Target Output**: The model is trained to predict the original masked spans in sequential order.
    - Target Output: <extra_id_0> brown fox <extra_id_1> over the lazy.
- This formulation forces the model to generate coherent text while learning contextual relationships between tokens.

T5 pre-training

- Predicting spans, rather than individual tokens, encourages the model to learn:
- **Global Context**: How spans relate to the larger sentence or paragraph structure.
- **Fluency and Cohesion**: Span prediction ensures generated outputs are natural and coherent.
- **Task Versatility**: The model is better prepared for downstream tasks like summarization, translation, and question answering.

T5 pre-training

- T5 pre-training uses the **C4** dataset (Colossal Clean Crawled Corpus), a massive dataset derived from Common Crawl.

- **Size**: Approximately 750 GB of cleaned text.

- **Cleaning**: Aggressive data cleaning is applied to remove spam, duplicate text, and low-quality content.

- **Versatility**: The dataset contains diverse text, helping the model generalize across domains.

T5 pre-training

- **Loss Function**: Cross-entropy loss is used for predicting masked spans.
- **Optimizer**: T5 employs the Adafactor optimizer, which is memory-efficient and designed for large-scale training.
- **Learning Rate Scheduling**: The learning rate is adjusted using a warm-up phase followed by an inverse square root decay.

T5 fine tuning

- **Input and Output as Text**: Fine-tuning continues the paradigm where the input and output are always text strings, regardless of the task.

- Example Tasks:
  - **Summarization**:
    - Input: summarize: <document> → Output: <summary>
  - **Translation**:
    - Input: translate English to French: <text> → Output: <translated_text>
  - **Question Answering**:
    - Input: question: <question> context: <context> → Output: <answer>

Popular T5 variants – mT5

- mT5 (Multilingual T5) was developed to extend T5's capabilities to multiple languages.
- It was pre-trained on the multilingual Common Crawl dataset covering 101 languages.
- Key Features:
  - Maintains the text-to-text framework across different languages.
  - No language-specific tokenization, since it uses SentencePiece with a shared vocabulary across languages.
  - Demonstrates strong multilingual performance, including on cross-lingual tasks.
- Applications:
  - Translation, multilingual summarization, and cross-lingual question answering.
- Limitations:
  - Larger model size due to the need to represent multiple languages in the vocabulary.
  - Performance can vary significantly across languages, favoring those with more representation in the training data.

Popular T5 variants – Flan-T5

- Flan-T5 is a fine-tuned version of T5 with instruction-tuning on a diverse set of tasks.

- Key Features:
  - Designed to improve generalization by training on datasets formatted as instruction-response pairs.
  - Better zero-shot and few-shot learning capabilities compared to the original T5.

- Applications:
  - Performs well in scenarios requiring generalization to unseen tasks, such as creative writing or complex reasoning.

- Limitations:
  - Requires careful task formulation to fully utilize its instruction-following capabilities.

Popular T5 variants – ByT5

- ByT5 (Byte-Level T5) processes text at the byte level rather than using subword tokenization.

- Key Features:
  - Avoids the need for tokenization, enabling better handling of noisy, misspelled, or rare words.
  - Works well for languages with complex scripts or low-resource scenarios.

- Applications:
  - Robust for tasks with noisy or unstructured text, such as OCR or user-generated content.

- Limitations:
  - Significantly slower and more resource-intensive due to longer input sequences (byte-level representation increases sequence length).

Popular T5 variants – T5-3B and T5-11B

- T5-3B and T5-11B are larger versions of the original T5 with 3 billion and 11 billion parameters, respectively.

- Key Features:
  - Improved performance on complex tasks due to increased model capacity.
  - Suitable for tasks requiring deep contextual understanding and large-scale reasoning.

- Applications:
  - Used in academic research and high-performance NLP applications where resources are not a constraint.

- Limitations:
  - Computationally expensive for fine-tuning and inference.
  - Memory requirements limit their usability on standard hardware.

Popular T5 variants – UL2

- UL2 (Unified Language Learning) is a general-purpose language model inspired by T5 but supports a wider range of pretraining objectives.

- Key Features:
  - Combines diverse learning paradigms: unidirectional, bidirectional, and sequence-to-sequence objectives.
  - Offers state-of-the-art performance across a variety of benchmarks.

- Applications:
  - General-purpose NLP tasks, including generation and comprehension.

- Limitations:
  - Increased complexity due to multiple pretraining objectives.

Popular T5 variants – Multimodal T5

- T5-Large Multimodal Variants combine T5 with vision capabilities by integrating additional modules for visual data.
- Key Features:
  - Processes both text and image inputs, enabling tasks like image captioning, visual question answering, and multimodal translation.
  - Often uses adapters or encodes visual features separately.
- Applications:
  - Multimodal tasks combining vision and language.
- Limitations:
  - Computationally expensive due to the need to process multiple modalities.

Popular T5 variants – Efficient T5

- Efficient T5 Variants are optimized for efficiency in resource-constrained environments.

- Examples:
  - **T5-Small/Tiny**: Reduced parameter versions of T5 for lower memory and compute needs.
  - **DistilT5**: A distilled version of T5, reducing the model size while retaining performance.

- Applications:
  - Real-time applications on edge devices or scenarios with limited computational resources.

- Limitations:
  - Sacrifices some performance compared to larger T5 models.

T5 variants

Variant	Purpose	Key Strengths	Limitations
mT5	Multilingual NLP	Supports 101 languages	Uneven performance across languages
Flan-T5	Instruction-following	Strong generalization	Needs task-specific prompts
ByT5	No tokenization	Handles noisy/unstructured text	Slower due to byte-level inputs
T5-3B/11B	High-capacity NLP	Exceptional performance	High resource requirements
UL2	Unified objectives	Versatility across tasks	Increased training complexity
Multimodal T5	Vision-language tasks	Combines text and image inputs	Higher computational cost
Efficient T5	Resource-constrained NLP	Lightweight, faster inference	Reduced task performance

Practice on translation and summarization

Practice

- Looking at the Hugging Face guides on translation https://huggingface.co/learn/nlp-course/chapter7/4?fw=pt and summarization https://huggingface.co/learn/nlp-course/chapter7/5?fw=pt , use various models to perform these tasks.

- By following the guides, if you have time and computational resources you can also fine tune one of the encoder-decoder models

Natural Language Processing and Large Language Models

Corso di Laurea Magistrale in Ingegneria Informatica

Lesson 15

Encoder-Decoder Transformers

Nicola Capuano and Antonio Greco

DIEM – University of Salerno