Large Language Models, Text Processing, and Audio Processing

Dr. Rajesh Kumar Mundotiya

August 14, 2025

Outline

1. Part 1: Text Processing - The Foundation
   - Tokenization
   - LLM Architectures and Training
2. Part 2: Extending to Speech Processing

Why Text Processing is Crucial

- Large Language Models, like all neural networks, operate on numbers, not raw text.
- This conversion is a critical first step that must capture:
  - Semantic meaning of words/tokens.
  - Syntactic structure and relationships.
  - Sequential order of the text.
- The process involves three key stages:
  1. **Tokenization**: Splitting text into smaller units.
  2. **Embedding**: Converting tokens into dense vectors.
  3. **Positional Encoding**: Adding information about word order.

From Text to Tokens

Definition

Tokenization is the task of splitting a sequence of text into smaller pieces called tokens. These tokens are then mapped to unique integer IDs from a vocabulary.

We will explore a few common strategies, starting with the most basic.

Tokenization Strategy 1: Word-based

- **Method:** Splits text by spaces and punctuation. It's the most intuitive approach.
- **Example:** `"LLMs are powerful." -> ["LLMs", "are", "powerful", "."]`

- **Advantage:**
  - Simple and easy to understand.

- **Disadvantage:**
  - **Huge Vocabulary Size:** Every unique word in a language requires an entry.
  - **Out-of-Vocabulary (OOV) Problem:** The model has no way to handle words it hasn't seen during training (e.g., "tokenization", new slang, misspellings).

Tokenization Strategy 2: Character-based

- **Method:** Splits text into its fundamental components: individual characters.
- **Example:** `"LLMs" -> ["L", "L", "M", "s"]`

- **Advantage:**
  - **Small Vocabulary:** Only need to store characters, numbers, and symbols.
  - **No OOV Problem:** Any word can be constructed from characters.

- **Disadvantage:**
  - **Very Long Sequences:** A single word becomes multiple tokens, increasing computational load.
  - **Loses Word-level Semantics:** The model has to learn the concept of a "word" from scratch.

The Best of Both Worlds

The Core Idea

Balance vocabulary size and sequence length by breaking rare words into smaller, meaningful subwords, while keeping common words as single tokens.

This handles new or complex words gracefully by composing them from known sub-parts.

Common Algorithms:

- **Byte-Pair Encoding (BPE):** Used by GPT series.
- **WordPiece:** Used by BERT.
- **SentencePiece:** Used by T5 and LLaMA.

The Full Transformer Block: Sub-layers

A single Transformer block (which is stacked many times) consists of two main sub-layers:

1. **Multi-Head Self-Attention (MHA) Layer:**
   - The mechanism we just discussed.
   - Allows the model to weigh the importance of different tokens in the sequence.

2. **Position-wise Feed-Forward Network (FFN):**
   - A simple two-layer fully connected neural network.
   - Applied independently to each position's representation after attention.
   - Adds expressive power and transforms the information.

$$\text{FFN}(\mathbf{x}) = \max(0, \mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2$$

The Full Transformer Block: Essential Additions

Two other components are crucial for making deep Transformers trainable:

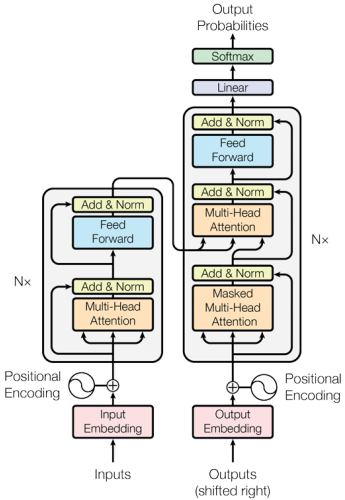
- **Residual Connections (or Skip Connections):**
  - The input to each sub-layer is added to its output.
  - This helps prevent the vanishing gradient problem, allowing for much deeper networks.

- **Layer Normalization:**
  - Applied after each sub-layer to stabilize the training process by normalizing the outputs.

The final operation for a sub-layer looks like this:

$$\mathbf{x}_{\text{out}} = \text{LayerNorm}(\mathbf{x}_{\text{in}} + \text{SubLayer}(\mathbf{x}_{\text{in}}))$$

The Transformer Block Architecture



LLM Architectural Flavors

By combining these blocks in different ways, we get three main families of LLMs. We'll look at each one.

1. **Encoder-Only**: For understanding content.
2. **Decoder-Only**: For generating content.
3. **Encoder-Decoder**: For transforming content.

Let's break them down.

Flavor 1: Encoder-Only (e.g., BERT, RoBERTa)

- **Architecture:** A stack of Transformer encoder blocks.
- **Attention:** Bidirectional. Each token can attend to all other tokens in the sequence (both past and future).
- **Training Objective:** Masked Language Modeling (MLM). Random tokens are hidden ("masked"), and the model must predict them based on the surrounding context.
- **Best For:** NLU (Natural Language Understanding) tasks like text classification, sentiment analysis, and named entity recognition.

Example: Text Classification with BERT

```
1  from transformers import pipeline
2
3  # Initialize a text classification pipeline with a pre-trained BERT model
4  classifier = pipeline('sentiment-analysis')
5
6  # Classify the sentiment of the sentence
7  result = classifier("I love ice cream!")
8  print(result)
9
10
11 # Output: [{'label': 'POSITIVE', 'score': 0.9996}]
```

Flavor 2: Decoder-Only (e.g., GPT, LLaMA, Mistral)

- **Architecture:** A stack of Transformer decoder blocks.
- **Attention:** Causal (or autoregressive). Each token can only attend to preceding tokens and itself. This prevents it from "seeing the future".
- **Training Objective:** Causal Language Modeling. The model is trained to predict the very next token in a sequence.
- **Best For:** NLG (Natural Language Generation) tasks like chatbots, text completion, and creative writing. This is the dominant architecture today.

Example: Text Generation with GPT-2

```
from transformers import pipeline

# Initialize a text generation pipeline with GPT-2
generator = pipeline('text-generation', model='gpt2')

# Generate text based on the prompt
result = generator("Once upon a time, in a faraway land", max_length=30,
                                                    num_return_sequences=1)
print(result[0]['generated_text'])


#Output : Once upon a time, in a faraway land, there lived a princess who
# had magical powers. She...
```

Flavor 3: Encoder-Decoder (e.g., T5, BART)

- **Architecture:** An encoder to process the source sequence and a decoder to generate the target sequence.
- **Attention:** Bidirectional in the encoder, causal in the decoder. The decoder can also attend to the encoder's output.
- **Training Objective:** Varies, often denoising or sequence-to-sequence mapping.
- **Best For:** Sequence-to-sequence tasks like machine translation, summarization, and question answering.

Example: Translation with T5

```
1  from transformers import pipeline
2
3  # Initialize a translation pipeline with T5
4  translator = pipeline('translation_en_to_fr', model='t5-base')
5
6  # Translate text from English to French
7  result = translator("Transformers are amazing!", max_length=40)
8  print(result[0]['translation_text'])
9
10
11 # Output: Les transformateurs sont incroyables!
```

Aligning the Model for Usefulness and Safety

The base model is powerful but not a helpful assistant. Fine-tuning adapts it.

- **Supervised Fine-Tuning (SFT) / Instruction Tuning:**
  - Train the base model on a smaller, high-quality dataset of (prompt, ideal_response) pairs curated by humans.
  - This teaches the model to follow instructions, be conversational, and adopt a specific persona.

- **Reinforcement Learning from Human Feedback (RLHF):**
  - Further refine the model's behavior by learning from human preferences.
  - Humans rank several model responses to a prompt, and a "reward model" is trained on this data. Reinforcement learning is then used to tune the LLM to maximize this reward.
  - Crucial for reducing harmful outputs and improving helpfulness.

Another Sequence, Another Modality

The principles we've discussed for text apply directly to other sequential data, like audio. The core challenge is the same: how do we convert a complex signal into numbers a model can understand?

For Text:

1. **Tokenization**
   (Text $\rightarrow$ Integers)
2. **Embedding**
   (Integers $\rightarrow$ Vectors)
3. **Modeling**
   (Transformer)

For Speech:

1. **Sampling**
   (Waveform $\rightarrow$ Digital Signal)
2. **Feature Extraction**
   (Signal $\rightarrow$ Spectrograms)
3. **Modeling**
   (Transformer)

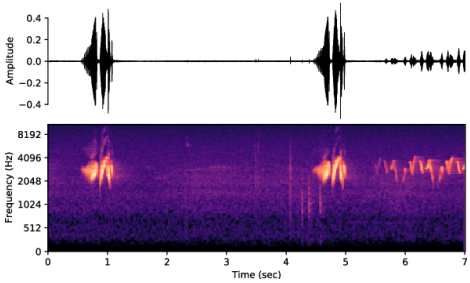
The Core Idea

We transform the raw audio waveform into a visual, information-rich representation that a Transformer can process, much like it processes text embeddings.

The "Tokenization" of Audio

A model can't directly process a raw audio waveform. We convert it into a **Mel Spectrogram**.

- **What it is:** A 2D "image" of sound, showing the intensity of different frequencies over time.
- **How it's made:** Using a Short-Time Fourier Transform (STFT) and then mapping frequencies to the Mel scale, which mimics human hearing.
- **Why it works:** It turns a complex 1D wave into a structured 2D representation, perfect for models that excel at finding patterns in data (like Transformers!).



Reusing Our Transformer Flavors

The same Encoder-Decoder architecture used for translation is perfect for speech tasks.

Automatic Speech Recognition (ASR): Audio → Text

- **Example Model:** OpenAI's Whisper
- **Encoder:** Processes the Mel Spectrogram ("listens" to the audio).
- **Decoder:** Autoregressively generates the corresponding text transcript.
- This is a classic sequence-to-sequence task.

Text-to-Speech (TTS): Text → Audio

- **Example Model:** Tacotron 2, VITS
- **Encoder:** Processes the input text embeddings.
- **Decoder:** Generates a Mel Spectrogram from the text representation.
- A separate model (a **Vocoder**) then converts this spectrogram back into a high-fidelity audio waveform.

Making Speech Models Practical

Speech models like Whisper are also extremely large and benefit immensely from quantization. All the concepts we just covered apply directly.

- **The Need:** Running ASR or TTS on-device (smartphones, smart speakers, cars) requires low latency and a small memory footprint.
- **The Solution:** Apply quantization to the model's weights and/or activations.
  - **Memory Savings (W4A16, GPTQ):** Allows a large, highly accurate speech model to fit on a consumer device.
  - **Speed Gains (W8A8):** Enables real-time, low-latency transcription or voice synthesis, crucial for a good user experience.
- **The Payoff:** Quantization is the key technology that moves state-of-the-art speech AI from the cloud to the edge, making it accessible everywhere.

References

- Attention Is All You Need https://arxiv.org/abs/1706.03762
- https://medium.com/@RobuRishabh/types-of-transformer-model-1b52381fa719
- https://www.researchgate.net/figure/Audio-spectrogram-representation-The-raw-audio-signal-is-transformed-using-the-Fourier_fig1_354098985

Questions?

Thank You!