Summary Report: Tokenization Fundamentals & GPT Architecture

Part 1: Tokenization Fundamentals

1. Subword Tokenization

- Breaks words into smaller units (subwords) to handle rare or unknown words.
- Balances vocabulary size and out-of-vocabulary tokens.

2. Byte Pair Encoding (BPE)

- Used in: GPT-2, GPT-3
- Merges the most frequent character pairs into subwords.
- Advantage: Efficient and handles rare words.
- Limitation: Merges are static after training.

3. WordPiece

- Used in: BERT
- Maximizes likelihood of subword sequences.
- Example: "unhappiness" -> [un, ##happy, ##ness]

4. SentencePiece

- Used in: T5, mBART
- Does not require pre-tokenized input; works on raw text.
- Supports BPE and Unigram LM.

Part 2: GPT Fundamentals

1. GPT Architecture

- Input Embedding: Converts token IDs into vectors.

- Positional Encoding: Adds order information to tokens.

- Transformer Decoder Blocks: Includes masked attention, feedforward layers, and LayerNorm.

- Output Head: Maps to vocabulary logits.

2. Self-Attention with Causal Masking

- Ensures each token attends only to previous tokens (no future leak).

3. Training Objective

- Causal Language Modeling: Predict the next token.

- Loss Function: Cross-entropy.

4. Text Generation

- Iteratively predicts and appends the next token.

- Sampling strategies: greedy, top-k, top-p (nucleus).

Comparison Table

Technique	Used In	Strength	Token Example
BPE	GPT-2	Merges frequent subword pairs	un, ##happy, ##ness
WordPiece	BERT	Probabilistic subword modeling	to, ##ken, ##izer
SentencePiece	T5, mBART	Language-independent tokenization	no whitespace needed

Why Tokenization Matters

- Affects how models interpret language.

- Impacts training speed, model generalization, and performance.