Core Concepts of Large Language Models (LLMs)

1. Introduction

Large Language Models (LLMs) such as GPT, Claude, and Gemini rely on a set of **core configuration concepts and parameters** that directly influence how responses are generated. Understanding these concepts is essential for **prompt engineering, application development, API integration, and responsible AI usage**.

This document explains the most important core concepts including **tokens, temperature, system roles, and other key parameters**, with examples and real-world software development use cases.

2. Tokens: The Fundamental Unit of LLM Processing

2.1 What Are Tokens?

A **token** is a chunk of text that an LLM processes. Tokens are **not the same as words**.

- A word may be split into multiple tokens
- Punctuation and spaces can be tokens
- Numbers and symbols are also tokenized

Example Tokenization

Text	**Tokens (Approximate)**
Hello	1 token
ChatGPT	1 token
ChatGPT is powerful	4–5 tokens
Internationalization	3–4 tokens

2.2 Why Tokens Matter

Tokens impact:

- Cost (API pricing is token-based)
- Context window size
- Performance and latency
- Prompt and response length limits

2.3 Input Tokens vs Output Tokens

Token Type	**Description**
Input tokens	Tokens sent in the prompt
Output tokens	Tokens generated by the model
Total tokens	Input + Output

Example

Prompt: Explain REST API (10 tokens)

Response: Detailed explanation (120 tokens)

Total usage: 130 tokens

2.4 Use Case: Token Awareness in Software Development

Scenario:

A chatbot integrated into a customer support system must handle long conversations.

Solution:

- Limit conversation history
- Summarize older messages
- Control max output tokens

---

3. Temperature: Controlling Creativity and Randomness

3.1 What Is Temperature?

Temperature controls how **random or deterministic** the model's output is.

Temperature Value	**Behavior**
0.0 – 0.2	Very deterministic, factual
0.3 – 0.5	Balanced
0.6 – 0.8	Creative
0.9 – 1.0	Highly creative, less predictable

3.2 Example

Prompt:

Generate a product description for a smartwatch

Temperature	**Output Style**
0.2	Technical and precise
0.7	Marketing-oriented and engaging
1.0	Very creative but may exaggerate

3.3 Best Practices

Use Case	Recommended Temperature
Code generation	0.0 – 0.3
Chatbots	0.4 – 0.6
Creative writing	0.7 – 0.9

---

4. System, User, and Assistant Roles

4.1 Role-Based Prompting

LLMs support **role-based messages** to control behavior.

Role	Purpose
System	Sets rules, tone, and behavior
User	Provides instructions or questions
Assistant	Model-generated response

4.2 System Role (Most Important)

The **system role** defines how the model should behave.

Example

System: You are a strict JSON API. Return only valid JSON.

User: Generate customer data.

Result:

- Output strictly follows structure
- No explanations added

4.3 Use Case: Enterprise Chatbot

System role:

You are a banking assistant.

Never provide financial advice.

Respond formally.

This ensures **compliance and consistency**.

---

5. Max Tokens: Limiting Response Length

5.1 What Is Max Tokens?

Defines the **maximum number of tokens** the model can generate in a response.

5.2 Why It Is Important

- Prevents long, unnecessary responses
- Controls cost
- Improves performance

5.3 Example

Max Tokens	Output
50	Short summary
300	Detailed explanation
1000	Long technical document

5.4 Use Case

Scenario:

Generating API error messages.

Solution:

Set max_tokens = 50 to avoid verbose outputs.

---

6. Top-p (Nucleus Sampling)

6.1 What Is Top-p?

Top-p limits token selection to the **most probable tokens whose cumulative probability ≤ p**.

Top-p Value	Effect
0.1	Very conservative
0.5	Balanced
0.9	More diverse

6.2 Temperature vs Top-p

Parameter	Controls
Temperature	Randomness
Top-p	Probability mass

Best Practice:

Use **either temperature or top-p**, not both aggressively.

---

7. Frequency and Presence Penalties

7.1 Frequency Penalty

- Reduces repeated words or phrases

7.2 Presence Penalty

- Encourages introducing new topics

7.3 Example

Without penalty:

This API is fast. This API is reliable. This API is scalable.

With penalty:

This API is fast, reliable, and scalable.

---

8. Practical Use Case: API-Based AI Assistant

Scenario

An organization builds an AI assistant for developers.

Configuration

Parameter	Value
System role	“You are a senior software architect”
Temperature	0.3
Max tokens	300
Top-p	0.9
Frequency penalty	0.2

Result

- Accurate
- Concise
- Professional responses

9. Common Mistakes to Avoid

- Ignoring token limits
- Using high temperature for code generation
- Not defining system role
- Mixing explanation with structured output
- Overusing long prompts unnecessarily

10. Summary Table of Core Parameters

Concept	**Purpose**	**Typical Use**
Tokens	Measure text size	Cost & limits
Temperature	Controls creativity	Style control
System role	Sets behavior	Governance
Max tokens	Output length	Cost control
Top-p	Probability filtering	Diversity
Penalties	Reduce repetition	Clean output

11. Conclusion

Understanding core LLM concepts such as **tokens, temperature, roles, and generation parameters** is critical for building reliable, cost-effective, and production-grade AI applications. These parameters allow developers to control **accuracy, creativity, safety, and consistency**, making LLMs suitable for real-world software systems.

When applied correctly, these concepts transform LLMs from simple chat tools into **powerful, configurable AI engines**.