# 5 Levels of Text Splitting Documentation

## 1. Introduction

Text splitting is a preprocessing technique used in Natural Language Processing (NLP) to divide large text into smaller, manageable chunks. This is especially important for applications such as Large Language Models (LLMs), document indexing, retrieval-based systems, and chatbots, where models have context-length limitations.

This project demonstrates **five progressive levels of text splitting**, starting from basic splitting and moving toward more intelligent, semantic-aware strategies.

---

## 2. Objective

- To understand why text splitting is required
- To explore multiple strategies for splitting text
- To compare simple and advanced splitting methods
- To prepare text for downstream NLP or LLM tasks

---

## 3. Why Text Splitting is Important

- Large documents exceed model context limits
- Improves retrieval accuracy in RAG systems
- Reduces memory and computation cost
- Preserves semantic meaning when done correctly

---

## 4. Level 1: Character-Based Splitting

### Description

- Splits text after a fixed number of characters
- Does not consider sentence or word boundaries

### Working

- Input text is divided every N characters
- Simple and fast method

### Advantages

- Easy to implement
- Works for uniform text size

### Limitations

- May break words or sentences
- Poor semantic preservation

---

## 5. Level 2: Word-Based Splitting

### Description

- Splits text based on word count
- Maintains complete words

### Working

- Text is tokenized into words
- Fixed number of words per chunk

### Advantages

- Better than character splitting
- Prevents word breakage

### Limitations

- Can still break sentences
- Limited semantic awareness

---

## 6. Level 3: Sentence-Based Splitting

### Description

- Splits text using sentence boundaries
- Uses punctuation as separators

### Working

- Text is divided at ‘.’, ‘?’, or ‘!’
- Each chunk contains complete sentences

### Advantages

- Preserves sentence meaning
- More readable chunks

### Limitations

- Chunk sizes may vary
- Still lacks topic awareness

---

## 7. Level 4: Recursive Text Splitting

### Description

- Hierarchical splitting strategy
- Uses multiple separators recursively

### Working

- Attempts splitting by paragraphs first
- Falls back to sentences or characters if needed

### Advantages

- Balanced chunk size
- Better structure preservation

### Limitations

- More complex implementation
- Slight computational overhead

---

## 8. Level 5: Semantic-Based Splitting

### Description

- Splits text based on meaning and context
- Uses embeddings or similarity measures

### Working

- Text is converted into vector embeddings
- Chunks are created when semantic similarity drops

### Advantages

- Best semantic preservation
- Ideal for RAG and search systems

### Limitations

- Computationally expensive
- Requires embedding models

---

## 9. Comparison of Text Splitting Levels

| Level | Method | Semantic Quality | Complexity |
|---|---|---|---|
| 1 | Character | Very Low | Very Low |
| 2 | Word | Low | Low |
| 3 | Sentence | Medium | Medium |

| Level | Method | Semantic Quality | Complexity |
|---|---|---|---|
| 4 | Recursive | High | High |
| 5 | Semantic | Very High | Very High |

---

## 10. Applications

- Retrieval-Augmented Generation (RAG)
- Chatbots and virtual assistants
- Document summarization
- Search engines
- Knowledge base indexing

---

## 11. Conclusion

This project demonstrates how text splitting evolves from simple mechanical techniques to intelligent semantic-aware strategies. Choosing the correct level depends on the application, performance requirements, and available computational resources.

---

## 12. Future Scope

- Hybrid semantic + recursive splitting
- Adaptive chunk size based on content
- Integration with vector databases