🔥 Stage 2 — Forward Index Construction (Document → Terms)

After building the **lexicon** in Stage 1, the next crucial step in a search engine pipeline is the **forward index**. This structure stores how each document maps to the words (terms) it contains. Forward indexes are essential for ranking, snippet generation, and phrase search.

📌 Why Stage 2 Matters

Imagine you type `"machine learning"` in a search engine:

1. Stage 1 (Lexicon) gives you **term IDs** for `"machine"` and `"learning"`.
2. Stage 2 (Forward Index) tells you:
    - Which **documents contain each term**.
    - How often each term appears in a document (**term frequency**).
    - Where the terms appear in the document (**positions**) for phrase searches.

Without this, you cannot score or rank documents efficiently.

📂 Forward Index Structure

Each document is represented as a record with:

Field	Description
doc_id	Unique document ID (0,1,2…N-1)
term_ids	List of term IDs in this document
term_freqs	How many times each term occurs in the document

positions	Position of each token in the document (optional)
length	Total number of tokens in the document

🧠 How It Works — Dry Run Example

Suppose your dataset has **2 documents**:

```
Doc0: apple banana apple
Doc1: banana mango
```

Lexicon Term IDs (Stage 1)

Term	Term ID
apple	0
banana	1
mango	2

Forward Index Representation

Doc0

- term_ids → `[0,1,0]` (apple, banana, apple)
- term_freqs → `[2,1,2]` (apple occurs 2 times, banana 1 time)
- positions → `[0,1,2]`
- length → `3`

Doc1

- term_ids → `[1,2]` (banana, mango)
- term_freqs → `[1,1]`

- positions → `[0,1]`
- length → `2`

📌 Why Segmentation Matters

For large datasets (thousands or millions of documents), storing all documents in a single file is inefficient.
Segmentation splits the forward index into smaller files, e.g., 1000 documents per segment. This:

- Speeds up read/write operations.
- Avoids rewriting huge files when new documents are added.
- Improves memory and disk efficiency.

📌 Key Advantages of Forward Index

1. **Efficient scoring**: Term frequencies are required for BM25 or TF-IDF ranking.
2. **Phrase search**: Positions allow exact phrase queries like `"apple banana"`.
3. **Snippet generation**: Helps highlight search terms in the original document text.
4. **Segmentation & compression**: Keeps index lightweight and scalable.

💡 Real-World Analogy

Think of Stage 1 (lexicon) as the **dictionary of words**, and Stage 2 (forward index) as **annotated notebooks** where each notebook is a document, showing exactly **which words appear, how often, and where**.

With both, a search engine can quickly answer queries without reading every document.

Next Step

- Stage 3 will build the **inverted index**:
    - Maps **term → list of documents** containing it.
    - Enables **fast search queries**, where the engine jumps straight to relevant documents.

✅ Summary

- Forward index stores **document-centric view** of terms.
- Includes **term IDs, frequencies, positions, and document length**.
- Segmented storage ensures **efficiency** on large datasets.
- Combined with lexicon, it prepares your search engine for **fast ranking and retrieval**.