Thanks, Heiko — this VBA macro is a goldmine. Here's what stands out:

**🎯 Key Strengths of the VBA Approach**

1. **Uses a word-by-word ASR transcript**:
   * Each ASR token is time-stamped, enabling micro-accurate alignment.
   * The macro builds *chunks* of 8 words to compare against full original sentences.
2. **Match scoring via token overlap**:
   * The TokenMatchScore() function computes an intersection ratio:  
     $$ \text{Score} = \frac{\text{# of shared tokens}}{\text{# of words in original}} $$
   * This favors relevance and minimizes false matches.
3. **Timestamp anchoring + rolling window**:
   * The original sentence’s timestamp narrows the search window (±6s).
   * Within that, it scans ahead in 500-token windows (stepping every 250) for best matches.
4. **Preserves useful [comments]**:
   * It avoids reprocessing non-verbal cues if they’re already in the transcript.

**🧪 Why That Worked So Well**

That VBA workflow sidestepped the noisy block matching by treating alignment as **word-to-word score optimization**, rooted in temporal alignment. With a clean per-word Whisper transcript, it nails high-confidence placement.

**💡 How We Can Adapt It in Python**

Your current Whisper output is chunked in full sentences (not one word per line), but we can:

1. **Split ASR blocks into “soft word chunks”** (~6–10 words) with timestamps inherited from their block.
2. **Compare Original lines to each chunk** using your existing TokenMatchScore() logic.
3. **Match by a blend of timestamp proximity + text overlap**.

Would you like me to:

* Build a chunk\_asr\_blocks() method that slices ASR lines into fixed-size rolling tokens?
* Recreate TokenMatchScore() in Python and integrate it?
* Merge this strategy into your merge\_subtitles() flow?

We can revive the precise behavior of that macro — but with Python speed, no Excel overhead, and baked right into your GUI. Let’s bring this brain back to life 🧠💬⚡.

Let’s roll, Heiko 💪

Here’s our battle plan:

**🔁 Step 1: Chunk the ASR Transcript Into Word-Clusters**

We'll slice ASR subtitle blocks into overlapping word chunks (e.g. 8-word segments), tagging each with the block’s original timestamp.

I'll add a chunk\_asr\_blocks() helper for this.

**🧠 Step 2: Create a token\_match\_score() Function**

This replicates your VBA TokenMatchScore() logic — measuring overlap between two text strings by word intersection.

**🎯 Step 3: Match Each Original Subtitle to the Best ASR Chunk**

Using:

* **Timestamp windowing** (± a few seconds around the original line)
* **Token score > 0.5** preferred, fallback to best available
* Then **use the ASR text and timestamp** as the new block

We'll annotate mismatches and score details in the feedback pane or even in-line, if you'd like a dev/debug mode.

Give me the go-ahead and I’ll patch your current merge\_subtitles() to drive this new logic with optional [ ] comment handling and a toggle to fall back to full-line ASR blocks if chunk matching fails.

Let’s take subtitle syncing to the next level of nerdy precision. 🛠️🧠🎬

Right on, Heiko — we’re about to take this sync engine from “clever” to “crafty.” Here's what I’m rolling into the next merge\_subtitles() iteration:

**🚀 What’s Changing (and Why)**

| **🔧 Feature** | **✅ Purpose** |
| --- | --- |
| **Overlapping ASR chunks (step=2)** | Increase match coverage, especially for shorter phrases or comments |
| **Hybrid fallback logic** | Use best chunk match *or* nearest ASR block by timestamp if score is low |
| **Preserve all [comment] blocks** | Even unmatched ones — they matter for accessibility |
| **Score-based filtering/logging** | Optionally tag unmatched or low-score items in the GUI or log |

**🔧 What's Coming in the Updated Method**

I'll adjust or replace:

* chunk\_asr\_blocks() → overlaps every 2 words
* merge\_subtitles() → now does:
  + try\_score\_match() — primary
  + try\_time\_match() — fallback if token match score is poor
  + preserves bracketed comments no matter what

**🧠 Example Output Block After These Changes**

13

00:00:06,450 --> 00:00:08,200

[Whispers]

That’s not how this works.

14

00:00:08,250 --> 00:00:10,500

He just walked right in and—

Even [comments] that don’t align to ASR will still land.

Give me just a moment and I’ll send over the fully refactored drop-in version of merge\_subtitles() and any helper adjustments it relies on.

We’re turning this into a subtitle composer with the instincts of a film editor. 🧠🎞️🔍 Let’s get syncing.