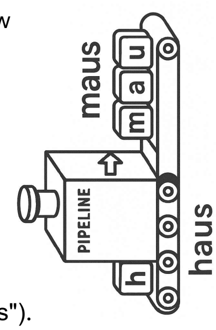
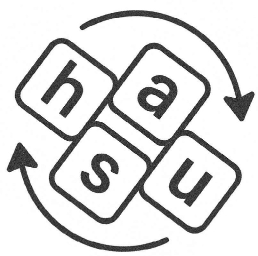
Edit-Distance Pipeline *(levenshtein.py)*

Refresh how the **four atomic edits** are mapped onto familiar PipelineSteps. You already know pipeline9.py; lean on it when questions about Pipeline behaviour arise.

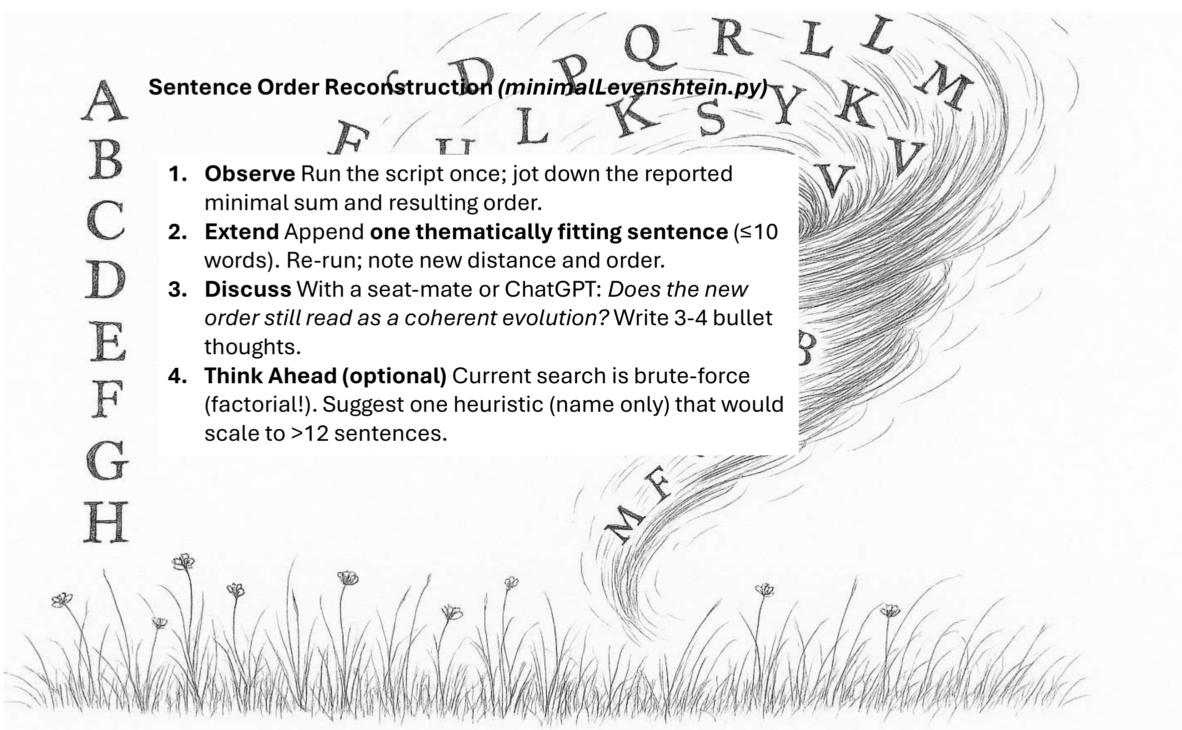


Mini-Glossary (fill in)

Op.	Symbol	One-word example	Result
Add	+x		
Del	-		
Swap	↔		
Move	→		



1. *Quick Trace:* Run levenshtein_distance("haus", "maus").
   - Insert a single print(idx, op) inside the inner loop;
   - stop when the first Move appears.
   - **Why is a rotation required?**
   - Add a one-sentence docstring in Move.process.
2. *Tiny Experiment:*
   - Replace the pair with your own **4-letter source/target** and confirm the produced pipeline still transforms correctly (assert).
3. *Reflect:*
   - Which edit is cheapest to omit if time is scarce and why?
   - How does treating the whole pipeline itself as a PipelineStep illustrate *composition*?



Sentence Order Reconstruction *(minimalLevenshtein.py)*

1. **Observe** Run the script once; jot down the reported minimal sum and resulting order.
2. **Extend** Append **one thematically fitting sentence** (≤10 words). Re-run; note new distance and order.
3. **Discuss** With a seat-mate or ChatGPT: *Does the new order still read as a coherent evolution?* Write 3-4 bullet thoughts.
4. **Think Ahead (optional)** Current search is brute-force (factorial!). Suggest one heuristic (name only) that would scale to >12 sentences.

Distance Trees *(distanceTrees.py)*

Compare how **different distance metrics** (Levenshtein, TF-IDF cosine, Jaccard token) influence hierarchical clustering of sentences.

1. **Observe** Run the script; three dendrogram windows appear. Fill the table (who merges first?):

Metric	First cluster (pair of sentence numbers)
Levenshtein	
Cosine TF-IDF	
Jaccard	

2. **Extend**: Add one sentence you expect to group with *“Die Sonne scheint heute hell.”* Re-run and note **which metric** placed it closest.
3. **Tweak:** For the *cosine* matrix change method in plot_dendrogram to 'complete' and 'ward' (two runs). Which linkage changes cluster heights the most?
4. **Reflect:** When is edit-distance *too literal* for textual clustering? Conversely, when might TF-IDF miss obvious thematic links?

Word Mover’s Distance *(wmd.py)*

Explore how Word Mover’s Distance quantifies semantic similarity between sentences and visualise its transport plan.

1. **Observe** Run the demo; jot down all three WMD values. Which sentence pair is semantically closest?
2. **Experiment** Insert a **fourth sentence** that you expect to be *very* close to Sentence 1. Re-run; record the new WMD values.
3. **Analyse** In the heatmap *Sentence 1 vs. Sentence 2*, mark the **two word pairs with the smallest distance**. Explain briefly why they are close.
4. **Compare (2–3 sentences)** Contrast WMD with simple word-overlap metrics (e.g. Jaccard). When might overlap fail?