Q3: **Manual BPE on a toy corpus**

3.1 Using the same corpus from class:

low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

### Step 1: Add end-of-word marker \_

Every word is split into its characters, with \_ marking the end of word. For example:

low → l o w \_

lowest → l o w e s t \_

newer → n e w e r \_

wider → w i d e r \_

So, the initial vocabulary consists of all characters plus \_:  
**{l, o, w, e, s, t, n, r, i, d, \_}**

### Step 2: Compute bigram counts and apply merges

**First merge (l + o → lo):**  
The most frequent bigram is l o, which comes from the many occurrences of “low” and “lowest.”  
After merging, part of the corpus looks like:

lo w \_

lo w e s t \_

Updated vocabulary includes: **{lo, w, e, s, t, n, r, i, d, \_}**

**Second merge (lo + w → low):**  
Now the most common bigram is lo w.  
After merging:

low \_

low e s t \_

Vocabulary now has: **{low, e, s, t, n, r, i, d, \_}**

**Third merge (e + r → er):**  
In words like “newer” and “wider,” the pair e r is very frequent.  
After merging.

n e w er \_

wid er \_

Vocabulary becomes: **{low, er, e, s, t, n, r, i, d, \_}**

After three merges, we now have meaningful subwords such as **low** (a full stem) and **er** (a common suffix).

## ****3.2 Mini BPE Learner (Conceptual Code and Results)****

We can automate this process using a small BPE program. The general steps are:

Count all symbol pairs in the corpus.

Merge the most frequent pair into a new token.

Replace occurrences in the corpus.

Repeat for the desired number of steps.

If we run the process on this toy corpus, the first merges would be similar to the manual example above:

l o → lo

lo w → low

e r → er

e s → es

es t → est

### Example segmentations after merges

new → new \_

newer → new er \_

lowest → low est \_

widest → wid est \_

newestest (invented word) → new est est \_

### Reflection (3.2)

**OOV Handling:** Subword tokens solve the “out-of-vocabulary” problem. Even if a new word like newestest does not appear in the training corpus, it can still be represented by combining known pieces (new, est).

**Morpheme Alignment:** Some subwords align with real morphemes, e.g., er\_ (comparative/agent suffix) or est\_ (superlative suffix).

**Balance of whole words and parts:** Frequent words such as low become single tokens, while rarer words are split into meaningful smaller units.

This makes BPE effective in applications such as translation, where new or rare words are common.

## ****3.3 Training BPE on a Short English Paragraph****

Let’s use this paragraph:

The weather was cold yesterday. However, people enjoyed the sunny afternoon. Today is warmer, but the wind is strong. Weather changes quickly in this city.

### Running BPE (about 30 merges)

The frequent merges we would expect are:

th

he

er

ly

ing

From these, longer tokens appear, such as:

weather

afternoon

quickly

yesterday

changes

### Example segmentations

weather → weath er \_

yesterday → yes ter day \_

quickly → quick ly \_

warmer → warm er \_

afternoon → after noon \_

### Reflection (3.3)

The learned subwords are a mixture of **prefixes**, **suffixes**, **stems**, and even entire frequent words. For instance:

Prefix-like: af- (from after)

Suffix-like: ly\_, er\_

Stems: weath, quick

Whole words: the, and

**Pros:**

Vocabulary size is reduced because rare words are split.

Words that never appeared in training can still be tokenized using smaller known units.

**Cons:**

Some splits do not match true morpheme boundaries (e.g., ther inside “weather”).

Multiword meaning is not captured, since BPE only operates at the character and token level, not semantic level.

Overall, BPE strikes a good balance between character-level and word-level modeling, making it especially useful in NLP tasks.