1 Task 1: BPE vs WordPiece Comparative Analysis

1.1 Early Merges & Morphological Learning

BPE merges are frequency-driven, with combinations like 'o' + 'n' → 'on' (216) and 't' + 'i' → 'ti' (186). It prioritizes high-frequency pairs and learns morphological units like 'tion' and 'ness'. However, BPE may split morphemes inconsistently. In contrast, WordPiece is score-driven and preserves meaningful morphemes earlier, using continuation markers (##) to keep affixes like '##ful' and '##ch' attached to the root. WordPiece focuses on subword patterns, such as '##e', '##i', '##t', and '##n', to better reflect common suffixes and roots.

1.2 Challenges

Both BPE and WordPiece face challenges when working with tokenization tasks. **Tie-breaking** issues arise when multiple pairs have identical frequencies or scores. BPE resolves this lexicographically, while WordPiece uses multi-level tie-breaking (score, frequency, and order), adding complexity but ensuring consistent merges. **Normalization**, such as lowercasing and punctuation separation, affects token distributions by creating high-frequency tokens from punctuation (like periods), reducing the model's ability to learn useful linguistic patterns. **Punctuation and Unicode handling** are also important while punctuation is separated to maintain meaning, this increases the token count. Both algorithms recompute pair frequencies after each merge. WordPiece requires scoring, which also increases computational cost. **Boundary Effects**: BPE uses explicit word boundaries (</w>) that can increase vocabulary pressure, while WordPiece uses implicit markers (##) to indicate subword continuation, preventing invalid cross-word merges.

BPE Training - First 10 merges:

Merge 1: 'o' + 'n' -> 'on' (frequency: 216)
Merge 2: 't' + 'i' -> 'ti' (frequency: 186)
Merge 3: 'e' + '</w>' -> 'e</w>' (frequency: 169)
Merge 4: 'e' + 'r' -> 'er' (frequency: 154)
Merge 5: 's' + '</w>' -> 's</w>' (frequency: 147)
Merge 6: 'e' + 'n' -> 'en' (frequency: 141)
Merge 7: 'y' + '</w>' -> 'y</w>' (frequency: 136)
Merge 8: 'ti' + 'on' -> 'tion' (frequency: 124)
Merge 9: 'a' + 'l' -> 'al' (frequency: 114)
Merge 10: 'e' + 's' -> 'es' (frequency: 98)

WordPiece Training - First 10 merges:

Merge 1: '##b' + '##j' -> '##bj' (score: 0.010753, frequency: 2)
Merge 2: 'o' + '##bj' -> 'obj' (score: 0.016393, frequency: 2)
Merge 3: 'e' + '##x' -> 'ex' (score: 0.009356, frequency: 9)
Merge 4: 'ex' + '##p' -> 'exp' (score: 0.011765, frequency: 9)
Merge 5: '##q' + '##u' -> '##qu' (score: 0.004348, frequency: 10)
Merge 6: 'q' + '##u' -> 'qu' (score: 0.004545, frequency: 3)
Merge 7: 't' + '##w' -> 'tw' (score: 0.003421, frequency: 13)
Merge 8: 't' + '##h' -> 'th' (score: 0.003256, frequency: 51)
Merge 9: 'w' + '##h' -> 'wh' (score: 0.003571, frequency: 48)
Merge 10: '##c' + '##k' -> '##ck' (score: 0.002687, frequency: 18)

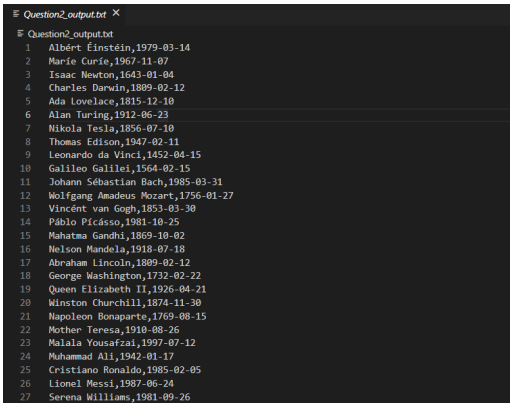
BPE Top-10 Tokens:

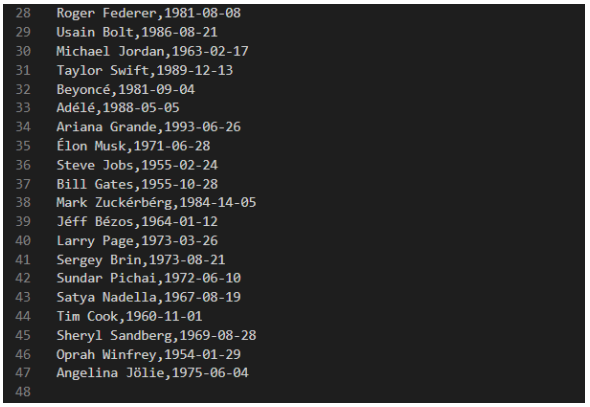
1. '</w>': 215 2. 'e': 173 3. 't': 166 4. 'o': 154 5. 'i': 145
6. 'p': 145 7. 'm': 144 8. 'c': 138 9. 's': 130 10. 'u': 123

WordPiece Top-10 Tokens:

1. '##e': 965 2. '##i': 705 3. '##t': 609 4. '##n': 598 5. '##o': 468
6. '##a': 455 7. '##r': 423 8. '##s': 422 9. '##l': 375 10. '##y': 191

Task 2: Regex Record Extraction





Regular Expressions Used

(i) Extract Records

r'\{([^{}]*)\}'

Purpose: Extracting content between curly braces {} while avoiding nested braces. Uses character class [^{}]* to match any character except braces.

(ii) Split/Classify Tokens - Date Detection

```
r'\b\d{1,2}/\d{1,2}/\d{2,4}\b'          # dd/mm/yyyy, mm/dd/yyyy formats
r'\b\d{4}[/-]\d{1,2}[/-]\d{1,2}\b'      # yyyy-mm-dd, yyyy/mm/dd formats
r'\b\d{1,2}-\d{1,2}-\d{2,4}\b'          # dd-mm-yyyy formats
r'\b\d{1,2}\.\d{1,2}\.\d{4}\b'          # dd.mm.yyyy formats
r'\b\d{1,2}-[A-Za-z]{3,9}-\d{4}\b'      # dd-Month-yyyy (15-Apr-1452)
r'\b\d{1,2}\s+[A-Za-z]{3,9}\s+\d{4}\b'  # dd Month yyyy (18 Jul 1918)
r'\b[A-Za-z]{3,9}\s+\d{1,2},\s*\d{4}\b' # Month dd, yyyy (January 15, 1929)
r'\b[A-Za-z]{3,9}\s+\d{1,2}\s+\d{4}\b'  # Month dd yyyy (December 18 2001)
```

Purpose: Comprehensive date detection covering slash, dash, dot separators and written month names with different word orders.

(iii) Phone Number Detection

```
r'\+\d+[-\s\(\)]*\d+[-\s\(\)]*\d+[-\s\(\)]*\d*'  # International: +country-area-number
r'\(\d+\)\s*\d+[-\s]*\d*'                        # Area code: (021) 34567890
r'\b\d{3,}[-\s]*\d*[-\s]*\d*\b'                  # Simple patterns: 124, 000-000-0000
r'\b\d+[-\s]\d+[-\s]\d+\b'                       # Complex: 92-21-1111111
```

Purpose: Identify phone patterns with international prefixes, parenthetical area codes, and various separator styles.

(iv) Name Validation

r'^[\w\s\.*\#\@\•\★\'\-\(\)àáâãäåèéêëìíîïòóôõöùúûüýÿñçÀÁÂÃÄÅÈÉÊËÌÍÎÏÒÓÔÕÖÙÚÛÜÝŸÑÇ]+$'

Purpose: Validate tokens as potential names, including Unicode accented characters for international names and common punctuation.

(v) Text Cleaning

```
r'[*#@•★\(\)\.]'  # Remove decorative punctuation from names
% r'\s+'          # Normalize multiple whitespace to single space
```

Purpose: Clean decorative symbols from names while preserving meaningful punctuation in dates/phones, and standardize whitespace.

Challenges Faced & Decisions Made

The sequence "7/11/67" presents ambiguity between European (dd/mm) and US (mm/dd) date formats, which was resolved by prioritizing date matching in the European format. Punctuation and whitespace were handled contextually, cleaning from names (e.g., "@Alan Turing #" → "Alan Turing") but preserving dates and phone numbers. Two-digit years were standardized with the rule: 00-24 → 2000-2024, 25-99 →1925-1999