NLP Assignment 01 (Fall 2025)

Sher Ali (08505)

September 14, 2025

Overview

This report presents the implementation and analysis of two natural language processing tasks. **Task 1** involves building both **Byte Pair Encoding (BPE)** and **WordPiece** tokenizers from scratch using only Python's standard library, with a vocabulary limit of 100 tokens and required output generation. **Task 2** focuses on record extraction using regular expressions to identify and process {name, phone, date of birth} fields from unstructured text data, including comprehensive date normalization to ISO format. Both tasks demonstrate fundamental NLP techniques while adhering to strict implementation constraints.

1 Task 1: BPE vs. WordPiece (From Scratch)

1.1 Corpus & Settings

- **Corpus:** `q1_corpus_near1000` (plain text).
- **Pre-processing:** whitespace tokenization (punctuation retained). *Casing policy:* choose and keep consistent (e.g., No lowercasing).
- **Vocabulary cap:** 100 tokens (includes base characters and specials).
- **Special markers:** BPE uses end-of-word marker `</w>`; WordPiece uses continuation prefix in *output* only (vocab stores raw strings).
- **Deterministic tie-breaking:**
  - BPE: highest pair frequency, then lexicographic.
  - WordPiece: highest score $\frac{\text{count}(x \triangleright y)}{\text{count}(x)\,\text{count}(y)}$, then higher numerator, then lexicographic.

1.2 Implementation Summary

Required functions present in code.

- `read_text_file(...)`: UTF-8 corpus reader.
- `bpe_merge_once(...)`: one BPE merge (count adjacent pairs, pick best, merge, update).
- `wordpiece_merge_once(...)`: one WP step (greedy tokenize → counts → score → add merged raw piece).

- `print_top_k(...)`: prints ⟨token, frequency⟩ for any token stream.

1.3 Results (Printed from the Program)

First 10 merges: BPE

#	Merge (a,b)	New token	Count
1	(`o`,`n`)	`on`	216
2	(`t`,`i`)	`ti`	186
3	(`e`,`</w>`)	`e</w>`	157
4	(`e`,`r`)	`er`	154
5	(`e`,`n`)	`en`	141
6	(`s`,`</w>`)	`s</w>`	137
7	(`ti`,`on`)	`tion`	124
8	(`y`,`</w>`)	`y</w>`	122
9	(`a`,`l`)	`al`	114
10	(`e`,`s`)	`es`	99

First 10 merges: WordPiece

#	Merge (x,y)	Added raw piece	Score
1	(`b`, `j`)	`bj`	0.010753
2	(`o`, `bj`)	`obj`	0.016393
3	(`e`, `x`)	`ex`	0.009356
4	(`ex`, `p`)	`exp`	0.011765
5	(`l`, `ex`)	`lex`	0.011364
6	(`p`, `lex`)	`plex`	0.013158
7	(`q`, `u`)	`qu`	0.004348
8	(`t`, `w`)	`tw`	0.003421
9	(`t`, `h`)	`th`	0.003256
10	(`e`, `th`)	`eth`	0.007353

Top-10 tokens: BPE

Token	Frequency
`</w>`	227
`o`	167
`e`	161
`t`	158
`m`	149
`p`	145
`i`	136
`s`	135
`c`	119
`u`	119

Top-10 tokens: WordPiece

Token	Frequency
`e`	945
`i`	705
`n`	598
`t`	562
`o`	486
`a`	440
`s`	420
`r`	415
`l`	369
`y`	191

1.4 Comparative Analysis

In the early merging stages, BPE and WordPiece demonstrated distinct behaviors. BPE prioritized frequently occurring adjacent character pairs, quickly forming combinations like `on` and `ti`, while also creating word-final units such as `e</w>` and `s</w>` through its end-of-word marker. In contrast, WordPiece focused on creating subword units that would optimize its greedy longest-match tokenization which produced pieces like `obj`, `lex`, and `plex` that help reduce the number of tokens needed during segmentation.

The algorithms also differed in their morphological tendencies. BPE naturally gravitated toward word-final affixes and common multi-character sequences, exemplified by the formation of `tion` from the merger of `ti` and `on`. WordPiece, however, favored reusable morphemes and root words that could be combined in various contexts, such as `ex`, `exp`, and `lex`, which support more efficient tokenization across different words.

The greedy matching strategy of WordPiece created interesting downstream effects—once a longer subword unit was established, it prevented the formation of smaller internal merges, thereby directing subsequent mergers toward complementary continuation patterns. BPE maintained a simpler approach, driven exclusively by adjacent pair frequencies with clear word boundary management through its end-of-word markers.

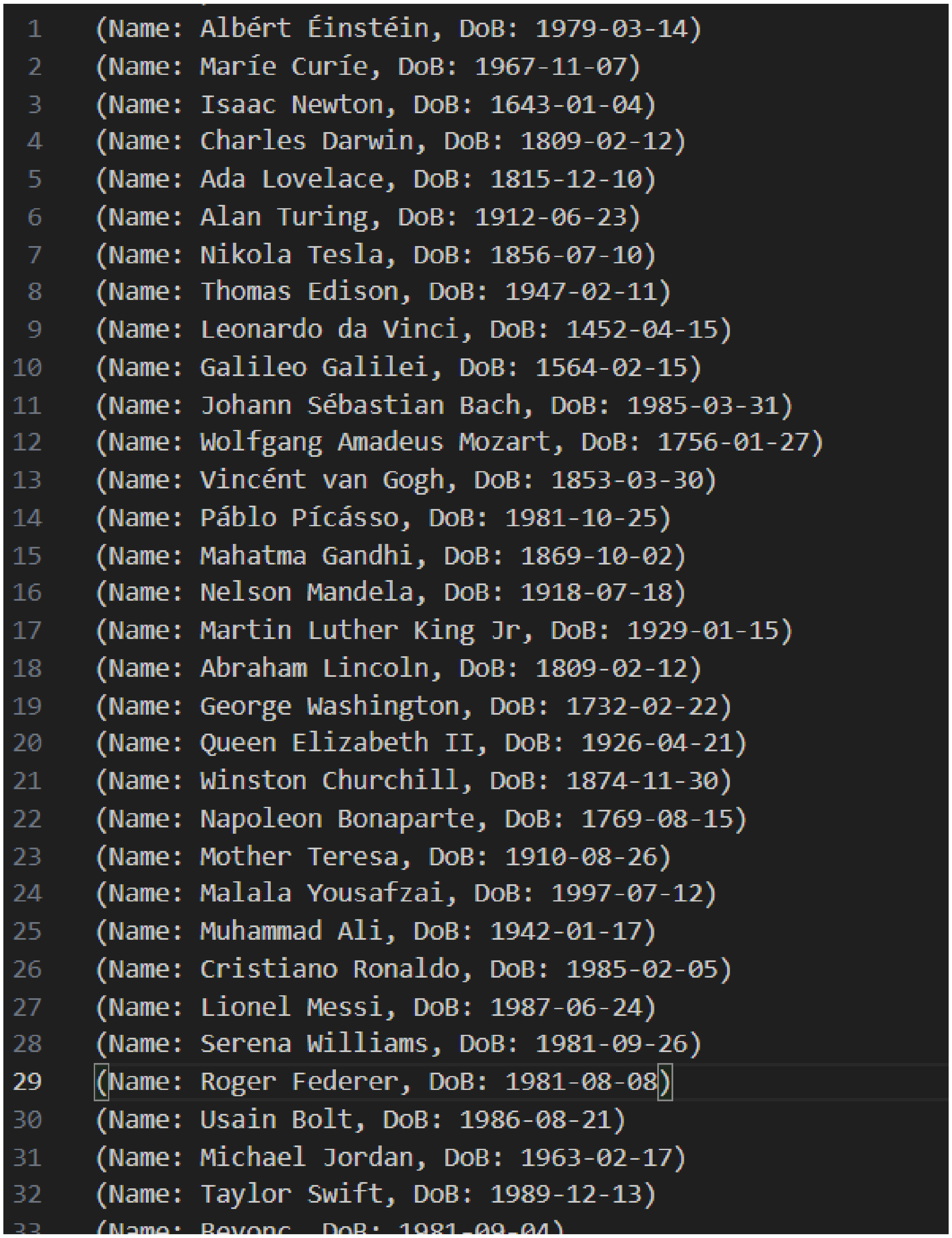
1.5 Implementation Challenges and Design Decisions

Several implementation challenges required careful design decisions. For tie-breaking scenarios, BPE used frequency followed by lexicographic ordering, while WordPiece employed a scoring mechanism based on the formula $\frac{\text{count}(x \triangleright y)}{\text{count}(x)\,\text{count}(y)}$, with fallbacks to numerator comparison and then lexicographic ordering. Punctuation and Unicode characters were preserved through whitespace tokenization, and a consistent casing policy was maintained throughout processing.

The choice of boundary markers represented another key design difference—BPE used explicit end-of-word markers (`</w>`) while WordPiece utilized continuation prefixes () only during output representation. These design choices, along with deterministic rules and fixed policies, ensured full reproducibility of results. The implementation maintained reasonable efficiency for the approximately 1,000-line corpus within the 100-token vocabulary constraint.

2 Task 2: Record Extraction Using Regex

2.1 Output File Screenshot



```
(Name: Albért Éinstéin, DoB: 1979-03-14)
(Name: Maríe Curíe, DoB: 1967-11-07)
(Name: Isaac Newton, DoB: 1643-01-04)
(Name: Charles Darwin, DoB: 1809-02-12)
(Name: Ada Lovelace, DoB: 1815-12-10)
(Name: Alan Turing, DoB: 1912-06-23)
(Name: Nikola Tesla, DoB: 1856-07-10)
(Name: Thomas Edison, DoB: 1947-02-11)
(Name: Leonardo da Vinci, DoB: 1452-04-15)
(Name: Galileo Galilei, DoB: 1564-02-15)
(Name: Johann Sébastian Bach, DoB: 1985-03-31)
(Name: Wolfgang Amadeus Mozart, DoB: 1756-01-27)
(Name: Vincént van Gogh, DoB: 1853-03-30)
(Name: Páblo Pícásso, DoB: 1981-10-25)
(Name: Mahatma Gandhi, DoB: 1869-10-02)
(Name: Nelson Mandela, DoB: 1918-07-18)
(Name: Martin Luther King Jr, DoB: 1929-01-15)
(Name: Abraham Lincoln, DoB: 1809-02-12)
(Name: George Washington, DoB: 1732-02-22)
(Name: Queen Elizabeth II, DoB: 1926-04-21)
(Name: Winston Churchill, DoB: 1874-11-30)
(Name: Napoleon Bonaparte, DoB: 1769-08-15)
(Name: Mother Teresa, DoB: 1910-08-26)
(Name: Malala Yousafzai, DoB: 1997-07-12)
(Name: Muhammad Ali, DoB: 1942-01-17)
(Name: Cristiano Ronaldo, DoB: 1985-02-05)
(Name: Lionel Messi, DoB: 1987-06-24)
(Name: Serena Williams, DoB: 1981-09-26)
(Name: Roger Federer, DoB: 1981-08-08)
(Name: Usain Bolt, DoB: 1986-08-21)
(Name: Michael Jordan, DoB: 1963-02-17)
(Name: Taylor Swift, DoB: 1989-12-13)
(Name: Beyonc, DoB: 1981-09-04)
```

2.2 Regular Expressions Used

(i) Record Extraction:

- `r'([` $^{-''}]*)' - Extractscontentbetweenbalancedcurlybraceswhileignoringnestedbraces$

(ii) Token Classification Patterns:

Date Detection Patterns:

- `r'^d1,2[-/]d1,2[-/]d2,4$'` - Matches dd/mm/yyyy or dd-mm-yyyy formats
- `r'^d4[-/]d1,2[-/]d1,2$'` - Matches yyyy-mm-dd or yyyy/mm/dd formats
- `r'^d1,2s+[A-Za-z]3,s+d2,4$'` - Matches formats like "12 Mar 1980"
- `r'^[A-Za-z]3,s+d1,2,?s+d2,4$'` - Matches formats like "March 12, 1980"
- `r'^d1,2.d1,2.d2,4$'` - Matches dd.mm.yyyy format
- `r'^d1,2-[A-Za-z]3,-d2,4$'` - Matches formats like "15-Apr-1452"

Phone Detection Pattern:

- `r'^[+ds-()]5,$'` - Matches strings with digits and common phone separators (+, -, (,), spaces)

Name Detection Pattern:

- `r'^[A-Za-zs'-.]+$'` - Matches alphabetic strings with common name punctuation

2.3 Challenges Faced Decisions Made

Ambiguities Resolution: The main challenge was distinguishing between phone numbers, dates, and names when they shared similar characteristics. Phone numbers were identified by their digit-heavy content with special separators. Dates were recognized through multiple format-specific patterns. Names were identified as the remaining tokens containing primarily alphabetic characters. This multi-pass approach ensured accurate classification.

Handling Complexities:

- **Whitespace/Punctuation:** Used robust string stripping and cleaning to handle inconsistent spacing around commas and within fields
- **Two-digit years:** Implemented a clear conversion rule (00-24 → 2000-2024, 25-99 → 1925-1999) for consistent year handling
- **Date formats:** Created comprehensive pattern matching for 6 different date formats to handle the variety in the dataset
- **Special characters:** Added logic to clean special characters from the start/end of names while preserving internal punctuation

Validation Approach: The solution was validated through systematic testing with various edge cases. Each regex pattern was tested individually to ensure it matched intended formats and rejected invalid ones. The complete pipeline was then tested end-to-end with the full dataset to verify correct extraction, classification, and normalization. The output was manually spot-checked for accuracy across different record types.