# Assignment 1: Text Cleaning and Basic Tokenization

Course: Computational Linguistics - 1

Deadline: January 27th, 2025 — 23:59

## 1 General Instructions

1. The assignment must be implemented in Python.
2. Only the re (regular expressions) library from Python's standard library should be used for text processing.
3. Submitted assignment must be your original work. Please do not copy any part from any source including your friends, seniors, and/or the Internet.
4. A single .zip file needs to be uploaded to the course portal.
5. Your grade will depend on the correctness of your implementation, the clarity of your code, and your understanding of regular expressions.
6. Please start early since no extension to the announced deadline would be possible.

## 2 Text Cleaning

Implement a text cleaning pipeline that processes raw text data. The pipeline should handle the following cases:

1. Basic Cleaning:
   (a) Remove extra whitespace (leading, trailing, and multiple spaces)
   (b) Convert text to lowercase
   (c) Remove special characters while preserving punctuation marks (e.g., .,!?)
   (d) Handle common abbreviations (e.g., "isn't", "don't", "I'm")
2. Advanced Cleaning:
   (a) Remove HTML tags (if present)
   (b) Replace of common tokens with Placeholders
      - Mail IDs with `<MAIL>`
      - URLs with `<URL>`
      - Hashtags with `<HASHTAG>`
      - Mentions with `<MENTION>`
   (c) Standardize quotation marks (both single and double quotes)

(d) Normalize different forms of ellipsis (..., . . ., . . . )

```
# Basic Cleaning Examples
Input: "  Hello   WORLD! "
Output: "hello world!"

# HTML Tag Removal
Input: "<p>This is a paragraph</p>"
Output: "This is a paragraph"

# Placeholder Substitution
Input: "Contact john.doe@example.com or @johndoe"
Output: "Contact <MAIL> or <MENTION>"

# Standardize quotation marks
Input: "She said 'hello' and "goodbye""
Output: "She said "hello" and "goodbye""
Convert mixed quote styles to consistent double quotes across the corpus.

# Ellipsis Normalization
Input: "I was thinking . . . maybe we should go."
Input: "I was thinking ..."
Input: "I was thinking . . ."
Output: "I was thinking..."
```

**Note:** This list is not extensive. You are also encouraged to try other tokenization and placeholder substitution schemes based on your observations from the corpora used. You're free to explore and add multiple such reasonable tokenization schemes in addition from the ones listed above. Implement these cleaning steps as separate functions that can be combined into a complete pipeline.

## 3 Basic Tokenization

Design a simple tokenizer using regular expressions that can handle the following cases:

1. Sentence-level tokenization:
   (a) Divide text into sentences
   (b) Handle various punctuations which are used as Sentence delimiter
2. Word-level tokenization:
   (a) Split sentences into words while preserving contractions
   (b) Handle hyphenated words appropriately
   (c) Properly separate punctuation from words
   (d) Identify and preserve common abbreviations (Mr., Dr., etc.)
3. Special cases (you may choose based your observations in corpus):

(a) Numbers (both cardinal and ordinal)

(b) Currency values ($100, €50)

(c) Dates (various formats)

(d) Time expressions (12:30, 2pm)

(e) Basic email addresses, Mentions, Mail IDs

## 4 Example Usage

Your implementation should handle the following examples correctly:

```
# Text Cleaning Example
Input: "  This is a   sample text with weird    spacing...  "
Output: "this is a sample text with weird spacing..."

# Tokenization Example
Input: "Mr. Smith's car cost \$15,000 in 2020!"
Output: ['Mr.', 'Smith's', 'car', 'cost', '\$15,000', 'in', '2020', '!']

# Sentence Tokenization
Input: "Hello! How are you? I'm fine."
Output: ["Hello!", "How are you?", "I'm fine."]

# Abbreviation Preservation
Input: "Dr. Smith met Mr. Johnson at 10 a.m."
Output: ['Dr.', 'Smith', 'met', 'Mr.', 'Johnson', 'at', '10', 'a.m.']

# Special Case Tokenization
Input: "Bought for $100 on 12/25/2020 at 2:30 pm"
Output: ['Bought', 'for', '$100', 'on', '12/25/2020', 'at', '2:30', 'pm']

# Hyphenated and Contracted Words
Input: "self-driving car isn't working"
Output: ['self-driving', 'car', 'is', 'not', 'working']
```

## 5 Dataset

Use the assigned text dataset from here

## 6 Submission Requirements

Submit a zip file named `<roll_number>_assignment1.zip` containing:

1. Source Code:

   (a) text_cleaner.py

   (b) basic_tokenizer.py

2. Documentation:
   (a) `README` specifying any assumptions
3. Test Results:
   (a) cleaned_text.txt containing cleaned text.
   (b) tokenized.txt containing tokenized text.

# 7 Resources

1. Python Regular Expressions Documentation:
   `https://docs.python.org/3/library/re.html`
2. Regular Expression Testing Tool:
   `https://regex101.com`
3. Lecture Slides