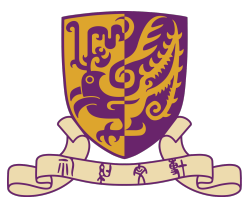
LING3401 Linguistics and Information Technology

Tutorial: Tokenization, word frequency analysis, and edit distance

Yige Chen

The Chinese University of Hong Kong

January 15, 2025



Acknowledgment

- Some of the tutorial materials are based on: Dan Jurafsky and James H. Martin. *Speech and Language Processing* (3rd ed. draft). 2024.

About Google Colaboratory (Colab)

- Google Colaboratory is a research tool for machine learning education and research. It's a Jupyter notebook environment that requires no setup to use.
- ...which means that you don't have to worry about setting up the Python environment on your device (at least for now)
- Today's Colab notebook has been posted

Tokenization

- Tokenization is the process of segmenting a text into tokens
  - Word tokenization: segmenting into words
  - Sentence tokenization: segmenting into sentences
  - ...
- A token is a single unit of text
  - ...and the basic unit for the processing of natural languages
  - Tokens can be words, subwords, numbers, punctuation, etc., based on tokenization strategies
- Example:
  - Text: Tokenization is the process of breaking down text into smaller units, such as words or sentences.
  - Word-level tokenization: [‘Tokenization’, ‘is’, ‘the’, ‘process’, ‘of’, ‘breaking’, ‘down’, ‘text’, ‘into’, ‘smaller’, ‘units’, ‘,’, ‘such’, ‘as’, ‘words’, ‘or’, ‘sentences’, ‘.’]

Tokenization

- Example:
  - Text: Tokenization is the process of breaking down text into smaller units, such as words or sentences.
  - Word-level tokenization: [‘Tokenization’, ‘is’, ‘the’, ‘process’, ‘of’, ‘breaking’, ‘down’, ‘text’, ‘into’, ‘smaller’, ‘units’, ‘,’, ‘such’, ‘as’, ‘words’, ‘or’, ‘sentences’, ‘.’]
- Seems easy, isn’t it?
- It looks like as long as we split texts on whitespace, we will be good?
  - Or, on whitespace and punctuation?
- Congratulations! You have developed a rule-based tokenizer!

Tokenization

- But how reliable is this rule-based tokenizer?
- Let's have a try!
- Example sentences
  - CUHK's Ph.D. program usually takes 4.5 years to complete.
  - 它一定也能給中文分詞吧!
    ('It must be able to segment Chinese tokens too!')
- What are the issues here?

Tokenization

- We need other rules!
- For English, we can specify the following:
  - When encountering an English clitic *'s*, treat it as a distinct token
  - For words like 'Ph.D.' and '4.5', do not split on punctuation
  - ...
- In this way, we got an improved rule-based tokenizer for English!
  - For those who are interested, this tokenizer can be implemented using regular expressions. Check out Chapter 2 of the recommended textbook!
  - (And of course, you don't need to know about this for the purpose of this course)

Tokenization

- And that's NLTK's tokenizer!
- Take a look at Part 1 of today's Colab notebook, and try out some English sentences yourself!

Tokenization

- But what if rules cannot handle complicated cases?
  - Think about languages like Chinese, Thai, Japanese, etc.
  - It's very difficult to come up with a complete set of rules to deal with languages without whitespace
- Can we train a tokenizer based on existing corpora and let the tokenizer itself figure out what should be tokenized?

Tokenization

- Then we have Byte-Pair Encoding (BPE)!
  - This is the tokenization strategy most commonly used by LLMs
- But before we dive into details, let me introduce some terms
  - Character (*computer sciences*): the smallest unit of text
    - e.g. the string 'language' has 8 characters: l, a, n, g, u, a, g, e
  - Vocabulary: the set of all unique tokens (words, subwords, or characters) used in a text or corpus
    - We will talk about this in detail a bit later
  - Subword tokens: words or parts of words or even individual letters

Tokenization

- How is BPE trained?
  1. Start with a vocabulary of all unique characters in the text
  2. Count the frequency of all adjacent pairs of symbols in the text
  3. Merge the most frequent pair into a single new symbol
  4. Repeat until the desired vocabulary size is achieved or no pairs are left
- Let's say that we have a text: *low lower lowest*, and we allow 2 merge steps
  - Initial vocab: 'l', 'o', 'w', 'e', 'r', 's', 't'
  - Most frequent adjacent pairs are 'lo': 3; 'ow': 3
  - Merge once: 'lo' added to vocab, now the most frequent adjacent pair is 'low': 3
  - Merge twice: 'low' added to vocab
- This is too technical so I consider it to be optional for this course! Just wanted to let you know how it works

Tokenization

- Take a look at Part 2 of today's Colab notebook, and see how a BPE tokenizer tokenizes texts differently!

Word frequency analysis

- Remember the term 'vocabulary' we introduced previously? This is firmly related to word frequency analysis!
- Vocabulary: set of unique words/tokens in the document/corpus/text, which we denote as V
- Word frequency: total occurrence of a word in the document/corpus/text
- Word types: number of distinct words/tokens in a corpus, i.e., $|V|$
- Word instances: the total number N of running words
- Type/Token Ratio (TTR): the number of types divided by the number of tokens (words), i.e., $\frac{|V|}{N}$

Word frequency analysis

- Take a look at Part 3 of today's Colab notebook, and try to calculate the word types, word instances, and TTR for the given text and for a text that you choose!
- Also, try to find the 5 most frequent words in the text!

Edit distance

- The minimum edit distance between two strings is defined as the minimum number of editing operations (operations like insertion, deletion, substitution) needed to transform one string into another
- For example, given the word *intention*:
  - Insertion: i n t e **c** n t i o n
  - Deletion: * n t e n t i o n
  - Substitution: i n **x** e n t i o n

Edit distance

- What is the minimum edit distance between *intention* and *execution*?

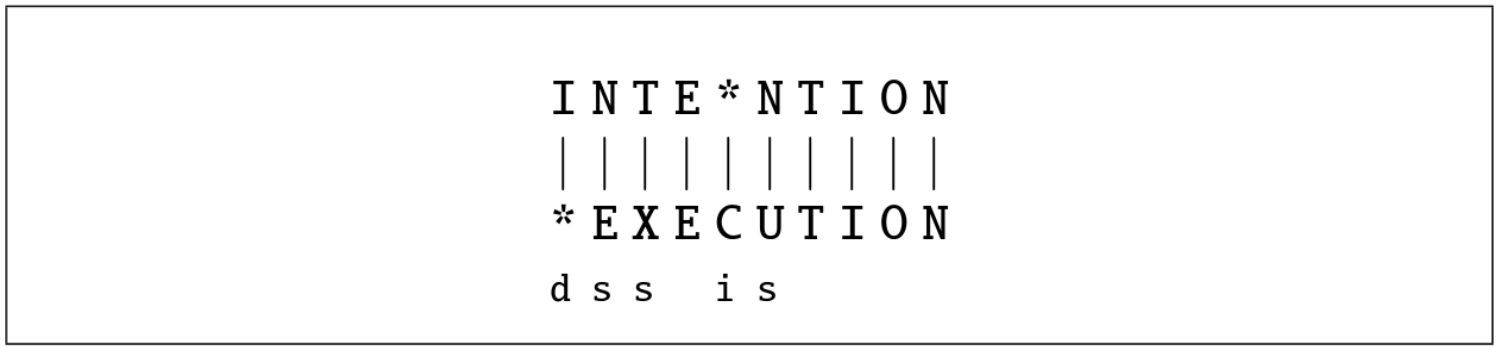


Figure 2.14 Representing the minimum edit distance between two strings as an **alignment**. The final row gives the operation list for converting the top string into the bottom string: d for deletion, s for substitution, i for insertion.

Edit distance

- Take a look at Part 4 of today's Colab notebook, and try to calculate the minimum edit distance between the two given words or words that you choose!
  - We will use NLTK's `edit_distance` function
- If you can, try to determine which editing operations are involved!

Miscellaneous

- Please do not hesitate to ask questions
- We enjoy feedback from you, so please let us know if you feel there's anything we could have done better
- It would be great if you'd bring your laptop to the class every week