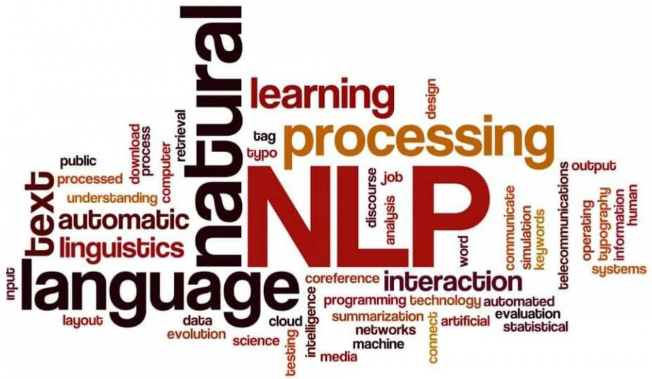
University of Northampton



Week 2

Traditional Linguistic Methods

CSY3055 – Natural Language Processing

Dr Oluseyi Oyedeji

Learning Goals for Week 2

By the end of week two, you should be able to:

- Use regular expressions (regex) for text cleaning and pattern extraction.
- Explain tokenisation at word, sentence, and subword levels.
- Compare stemming and lemmatisation, with their strengths and limits.
- Describe segmentation in languages without spaces.
- Discuss how tokenisation can create bias in multilingual NLP.

Relevance

- Every NLP system, from Google Translate to ChatGPT, starts with these **basic steps**.
- If text is cleaned or segmented wrongly, **errors** flow into the whole pipeline.
- Regex, tokenisation, and segmentation lets **NLP systems** flow seamlessly

If AI is "smart", why do we still need these simple rules?

Today's Session

- Regular Expressions (Regex)
- Tokenisation
- Ethics
- Stemming & Lemmatisation
- Segmentation

University of Northampton

Regular Expressions (Regex)

What is a Regular Expression (Regex)?

- A regex is a **pattern** that matches text.
- Example:
  - `\d{3}-\d{2}-\d{4}` matches a U.S. Social Security Number.
- In NLP
  - regex helps us clean noisy text and extract useful pieces.

Regex and Finite Automata

- Every regex = a **finite automaton** (a small machine that accepts or rejects strings).
- Example: [a-z]+ → "match one or more lowercase letters".

$$L(r) = \{w \in \Sigma^* \mid M_r \text{ accepts } w\}$$

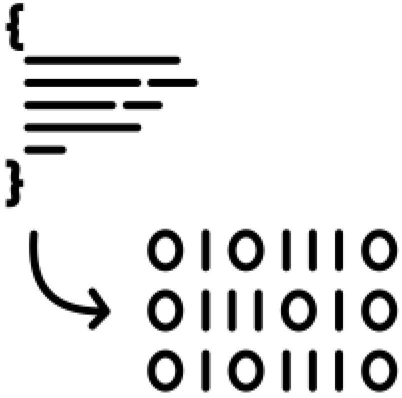
  - the set of all strings (w) that match regex (r).

- When you run regex, you're actually running a machine from automata theory!

University of Northampton

How Regex Becomes a Machine

- Thompson's Construction builds a **nondeterministic finite automaton (NFA)** from **regex**.
- Example: `(ab|c)*` creates a machine that branches between ab or c.
- Many tools (like **compilers**, **search engines**) rely on this principle.



Efficiency & Pitfalls

- **Deterministic Finite Automaton (DFA)** = fast, linear time.
- Backtracking engines (like Python's re) can explode to exponential time.
- **Danger: (a+)+b** → takes forever on long strings of a.
- This is called **catastrophic backtracking.**
- Why do we still use backtracking engines if DFA is faster?

University of Northampton

Regex in NLP Tasks

- Cleaning text
  - **remove URLs, HTML tags, punctuation.**
- Extracting info
  - **emails, hashtags, dates, phone numbers.**
- Bootstrapping Named Entity Recognition (NER)
  - **first pass with regex before ML.**

Code Demo: Regex in Python

```python
import re
text = "Email: jane.doe@uni.edu #NLP"
emails = re.findall(r"\b[\w\.-]+@[\w\.-]+\.\w+\b", text)
tags = re.findall(r"(?<!\w)#[\w]+", text)
print(emails, tags)
```

Output: ['jane.doe@uni.edu'] ['#NLP']

Real NLP Applications of Regex

- Clinical NLP
  - extract patient IDs from medical notes.
- Social media analysis
  - find hashtags, mentions.
- Legal NLP
  - locate contract dates, financial codes.

Regex is still essential, even in the deep learning era.

Limitations of Regex

- **Brittle**
  - Fails on variation (“3 Oct” vs “October 3rd”).
- Cannot capture **nested structure** (e.g., parentheses, long-distance rules).
- Regex works **best as a first filter**, **not the full NLP solution**.

Tokenisation

What is Tokenisation?

- **Tokenisation** means **breaking text** into units (tokens).
- Example :

  “**I’m happy**” → [I, ’m, happy].

- Foundation for **search engines**, **translation**, **language models**.

University of Northampton

Tokenisation as Segmentation

- **Problem** is how to **split input (x)** into **best tokens (s).**

$$s^* = \arg\max_{s \in \mathcal{S}(x)} P(s|x)$$

pick the **segmentation (s)** that is *most probable for input (x).*

Do humans also “tokenise” when reading?

Word Tokenisation Challenges

- **Spaces** are not enough!
- Problems:
  - **Contractions**: “don’t → do + n’t”
  - **Hyphens**: “state-of-the-art”
  - **Numbers**: “3,000.5”
- Tools like **spaCy** and **NLTK** handle these rules.

Sentence Tokenisation

"Dr. Smith went home. He slept."

- Is "**Dr.**" the end of a sentence? No.
- **Punkt Algorithm** learns abbreviation **patterns** and **sentence breaks**.

Why Subword Tokenisation?

- **Infinite words**; we cannot store them all.
- *Subwords* solve the out-of-vocabulary (OOV) problem.
- Examples:

"blockchain" → block + chain.

"unhappiness" → un + happi + ness.

Byte Pair Encoding (BPE)

- **Algorithm**
  - Start with characters.
  - Merge most frequent pair.
  - Repeat until vocab size reached.

$$(a^*, b^*) = \arg\max_{(a,b)} \mathrm{freq}(ab)$$

merge the letters that appear together the most.

- GPT tokenisers use this method.

WordPiece & Unigram LM

- **WordPiece** chooses **segmentation** that *maximises sequence probability*.

$$s^* = \arg\max_{s} \prod_{w \in s} P(w)$$

pick the split that makes the sentence most likely.

- **Unigram LM** is a *probabilistic model* trained via **Expectation-Maximisation (EM).**

Tokenisation in Real NLP Systems

- **Search engines** to *index keywords*.
- **Machine Translation** for *word alignment* across languages.
- Large Language Models (**LLMs**)
  - BERT = WordPiece.
  - GPT = BPE.
  - T5 = SentencePiece.

If GPT uses subwords, why does it sometimes cut words in funny places?

Code Demo: spaCy Tokeniser

```
import spacy
nlp = spacy.load("en_core_web_sm")
doc = nlp("Dr. Smith’s AI-based model is state-of-the-art.")
print([t.text for t in doc])
```

Output shows clean tokens, even with tricky hyphens.

Stemming vs Lemmatisation

Stemming

- **Stemming** is the process of *reducing inflected form* of a word to one so-called “**stem**,” or **root form**
- **Rule-based truncation** of words.
- Example:

  “caresses → caress”

  “ponies → poni”.

- **Fast**, but *not always meaningful*.

University of Northampton

Lemmatisation

- **Lemmatisation** is used to break a word down to its root meaning to identify similarities.
- **Lemmatisation** uses dictionary with morphological analysis.
- Example

“better → good”

“ate → eat”

- **Slower**, but *linguistically correct*.

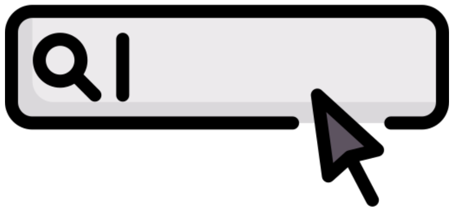
Lemmatisation as Normalisation

$$f(w, pos) \mapsto \ell$$

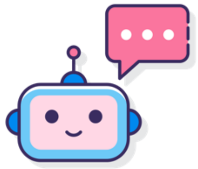
Given a ***word (w)*** and its ***part of speech (pos)***, return the base **lemma (l).**

NLP Applications

- **Stemming**
  - Used in search engines to increase recall.



- **Lemmatization**
  - Used in chatbots, MT, sentiment (preserves meaning).



- **Example**
  - Sentiment system should know

"better = good", not "bet".

Code Demo: NLTK

```
from nltk.stem import PorterStemmer, WordNetLemmatizer
ps, wnl = PorterStemmer(), WordNetLemmatizer()
words = ["studies", "studying", "better", "ate"]
print([(w, ps.stem(w), wnl.lemmatize(w, "v")) for w in
words])
```

Limitations

- Stemming
  - Can be **unrefined** (university → univers).
- Lemmatization
  - needs **large lexicons**.
- Trade-off
  - **speed vs accuracy**.

When would you prefer speed over accuracy in NLP?

Segmentation

Why Segmentation?

- Languages like **Chinese**, **Japanese**, **Thai** do not use spaces.
- Without segmentation, it would be impossible to process certain texts

e.g. "**我喜欢学习**"

Maximum Matching

- Maximum matching algorithm
  - Performs segmentation by greedily taking the **longest word** found in the dictionary
- Greedy
  - take longest **dictionary word**.
- Fast, but ambiguous.

Statistical Segmentation (Viterbi)

- **Statistical segmentation** uses **probability models** to decide where to split words in a sentence.
- Use probabilities to choose best cut.

$$V[t] = \max_{k \leq L} \{ V[t-k] + \log P(x_{t-k+1:t}) \}$$

At each position, look back up to **(L) characters** and pick the split that makes the sentence **most probable**.

University of Northampton

Example (Chinese)

- Sentence

我喜欢学习

- Possible segmentations

[我, 喜欢, 学习] = "I / like / study" - **yes**
[我, 喜, 欢学, 习] = – **no**

- Probability model decides correctly.

Conditional Random Field (CRF)

- CRF = Conditional Random Field
- CRF is a model for sequence labelling.
- **Labels**
  - B = Begin, M = Middle, E = End, S = Single.

$$P(y|x) = \frac{\exp(\sum_i \theta \cdot f(y_{i-1}, y_i, x, i))}{Z(x)}$$

score all possible label sequences, normalise them, pick the best one.

University of Northampton

Code Demo: Jieba Segmentation

```
import jieba
print(list(jieba.cut("我喜欢学习")))
```

Output: ['我', '喜欢', '学习']

Ethics & Bias

University of Northampton

Token Inflation

- English

  "**I love NLP**" → 3 tokens.

- Chinese:

  "**我喜欢自然语言处理**" → ~10 tokens.

- Same meaning, more tokens = unfair cost.

Impacts of Tokenisation Bias

- **Higher API costs** for some languages.
- **Shorter** usable context in **LLMs** for **token-heavy scripts**.
- Models **favour languages** with **more efficient tokenisation** (often English).

Towards Fairer Tokenisation

- **Masakhane (African NLP group)** → work on better tokenisers for African languages.
- Research question
  - *Can we design tokenisers that treat languages equally?*
- Ethics in NLP is not only about biased data, but also about biased infrastructure.

Practical & Wrap-Up

Summary

- Regex is a quick pattern matcher, but brittle.
- Tokenisation is the foundation of all NLP pipelines.
- Stemming vs Lemmatization = speed vs meaning.
- Segmentation is mandatory for non-spaced languages.
- Ethics = tokenisation choices create real-world unfairness.
- How would you design a tokeniser for non-English languages?

Thank you