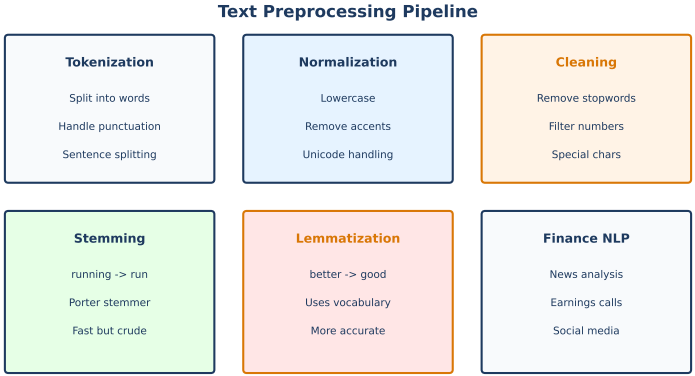
Lesson 37 Summary: Text Preprocessing

Data Science with Python – Key Concepts

Data Science Program

Text Preprocessing Overview



Clean text is essential for NLP success

Tokenization

First step: split text into units

- **Word tokens:** Split on whitespace/punctuation
- **Sentence tokens:** Split on sentence boundaries
- **Subword:** BPE for rare words

Use nltk.word_tokenize() or spacy

Normalization

Standardize text format:

- **Lowercase:** Reduce vocabulary size
- **Unicode:** Normalize accents
- **Numbers:** Replace with tokens

text.lower() is simple but effective

Stopword Removal

Remove low-information words:

- **Common:** the, a, is, and, of
- **Domain-specific:** Add custom stopwords
- **Caution:** Not always beneficial

```
from nltk.corpus import stopwords
```

Stemming vs Lemmatization

Reduce words to root form:

- **Stemming:** Fast, rule-based (running $\rightarrow$ run)
- **Lemmatization:** Accurate, uses vocabulary
- **Trade-off:** Speed vs accuracy

Lemmatization preferred for understanding

Regular Expressions

Pattern matching for cleaning:

- **Remove URLs:** http pattern matching
- **Extract:** Emails, numbers, dates
- **Replace:** Normalize patterns

---

import re; re.sub(pattern, replacement, text)

Quick Reference

Essential Commands:

Task	Code
Tokenize	`nltk.word_tokenize(text)`
Lowercase	`text.lower()`
Stopwords	`nltk.corpus.stopwords.words('english')`
Stem	`PorterStemmer().stem(word)`
Lemmatize	`WordNetLemmatizer().lemmatize(word)`

Preprocessing quality determines model quality