Module 03: NLP & Sentiment Analysis — Cheat Sheet

ML for Innovation Research — Prof. J. Osterrieder

Text Preprocessing Pipeline

1. **Tokenise** — split text into words.
2. **Lowercase** — normalise case.
3. **Remove stopwords** — drop "the", "is", "and" ...
4. **Stem / Lemmatise** — reduce to root form.
   Stemming: "running" → "run" (rule-based).
   Lemmatisation: "better" → "good" (dictionary-based).
5. **Vectorise** — convert tokens to numbers.

VADER Sentiment Analysis

Rule-based lexicon for social-media-style text.

Returns four scores per document:

Score	Meaning
`pos`	Proportion positive
`neg`	Proportion negative
`neu`	Proportion neutral
`compound`	Normalised aggregate (-1 to $+1$)

Compound thresholds:

- $\geq +0.05 \rightarrow$ **Positive**
- $\leq -0.05 \rightarrow$ **Negative**
- Between $\rightarrow$ **Neutral**

TF-IDF

Term Frequency – Inverse Document Frequency

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

$$\text{IDF}(t) = \log \frac{N}{|\{d : t \in d\}|}$$

$\text{TF}(t, d)$	frequency of term t in document d
N	total number of documents
$\text{IDF}(t)$	rarity of term across corpus

- High TF-IDF = term is frequent locally but rare globally.
- Common words get *down-weighted* automatically.
- Output: sparse document–term matrix.

Key Code Snippets

```
from nltk.sentiment.vader import \
    SentimentIntensityAnalyzer
from sklearn.feature_extraction.text \
    import TfidfVectorizer
import nltk
nltk.download('vader_lexicon')

# VADER sentiment
sia = SentimentIntensityAnalyzer()
scores = sia.polarity_scores(
    "Great product, highly recommend!")
# {'neg':0, 'neu':0.36,
#  'pos':0.64, 'compound':0.76}

# TF-IDF
tfidf = TfidfVectorizer(
    max_features=500,
    stop_words='english')
X_tfidf = tfidf.fit_transform(docs)
feature_names = tfidf.get_feature_names_out()
```

Preprocessing Code

```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

tokens = word_tokenize(text.lower())
stops = set(stopwords.words('english'))
filtered = [t for t in tokens
            if t.isalpha() and t not in stops]
stemmer = PorterStemmer()
stemmed = [stemmer.stem(t) for t in filtered]
```

When to Use What

- **VADER:** quick sentiment polarity; no training needed.
- **TF-IDF:** feature extraction for downstream ML (clustering, classification, topic modelling).
- For domain-specific text, consider a fine-tuned model or manual lexicon adjustments.