Module 04: Topic Modelling & Synthesis — Cheat Sheet

ML for Innovation Research — Prof. J. Osterrieder

Latent Dirichlet Allocation (LDA)

Intuition: Every document is a mixture of topics; every topic is a mixture of words.

1. For each document, sample a topic distribution.
2. For each word in the document:
   a. Pick a topic from the document's distribution.
   b. Pick a word from that topic's word distribution.
3. Learn topic–word and document–topic matrices via inference.

Key inputs:

- Document–term matrix (from TF-IDF or CountVectorizer).
- k = number of topics (you choose).

Choosing the Number of Topics

Metric	Guidance
Perplexity	Lower is better; measures held-out likelihood.
Coherence (C_v)	Higher is better; measures semantic consistency of top words.
Human review	Read top-10 words per topic—do they make sense?

Typical approach: sweep $k \in \{3, 5, 7, 10, 15\}$ and pick the k with the best coherence that is still interpretable.

Cross-Tabulation: Clusters × Topics

After running both K-Means and LDA independently:

1. Assign each document a **cluster** (K-Means).
2. Assign each document a **dominant topic** (LDA).
3. Build a contingency table (cross-tab).
4. Look for patterns: do certain clusters align with certain topics?

This triangulation strengthens your findings by showing that structure found via numerical features (clusters) aligns with structure found via text (topics).

Combining Unsupervised Methods

The synthesis workflow:

1. **Cluster** on numerical features → groups.
2. **Profile** each cluster (means, distributions).
3. **Topic-model** text fields → themes.
4. **Cross-tabulate** clusters × topics.
5. **Sentiment analysis** per cluster/topic.
6. **Validate** — do combined insights tell a coherent story?

Key Code Snippets

```
from sklearn.decomposition import \
    LatentDirichletAllocation
from sklearn.feature_extraction.text \
    import CountVectorizer

# Create document-term matrix
cv = CountVectorizer(max_features=1000,
                     stop_words='english')
dtm = cv.fit_transform(docs)

# Fit LDA
lda = LatentDirichletAllocation(
    n_components=5, random_state=42)
lda.fit(dtm)

# Top words per topic
for i, topic in enumerate(lda.components_):
    top = topic.argsort()[-10:][::-1]
    words = [cv.get_feature_names_out()[j]
             for j in top]
    print(f"Topic {i}: {', '.join(words)}")

# Dominant topic per document
topic_dist = lda.transform(dtm)
dominant = topic_dist.argmax(axis=1)
```

Synthesis Tips

- Always label clusters and topics with *human-readable* names.
- Use heatmaps to visualise the cross-tabulation.
- Report both quantitative metrics *and* qualitative interpretation.