Topic Modeling in R: LDA vs NMF with Associated Press

Contents

Overview & Learning Goals

This R Markdown file presents a comparison of two topic modeling approaches using an R dataset:

- **Latent Dirichlet Allocation (LDA)** — a probabilistic model that learns topics as word distributions and documents as mixtures of topics.
- **Non-negative Matrix Factorization (NMF)** — a linear-algebra approach that factorizes a document–term matrix into non-negative parts ("topics") and document weights.

We will:

1. Use the **Associated Press (AP)** corpus bundled in the `topicmodels` package.
2. Build a **shared vocabulary** once and reuse it for both models.
3. Fit **LDA on counts** and **NMF (KL divergence) on V__counts** computed from the same counts.
4. Inspect **top words** per topic.
5. Compute two **model-agnostic** metrics:
   - **UMass coherence** (higher/less negative ≈ better internal consistency).
   - **Mean Jensen–Shannon distance (JSD)** between topics (higher = more distinct).

Setup

Packages

We use `topicmodels` (LDA + AP dataset), `NMF` (NMF), and base sparse-matrix tooling via `Matrix` and `slam`.

```r
pkgs <- c("topicmodels", "NMF", "Matrix", "slam")
to_install <- setdiff(pkgs, rownames(installed.packages()))
if (length(to_install)) install.packages(to_install, repos = "https://cloud.r-project.org")

# NMF depends on Biobase (from Bioconductor). Install if missing.
if (!requireNamespace("Biobase", quietly = TRUE)) {
  if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")
  BiocManager::install("Biobase", ask = FALSE, update = FALSE)
}
if (!requireNamespace("tm", quietly = TRUE)) install.packages("tm")
library(tm)

library(topicmodels)
library(NMF)
library(Matrix)
library(slam)
set.seed(0)
```

Configuration

```r
n_topics     <- 10         # number of topics
n_top_words  <- 20         # words to display per topic
max_features <- 5000       # vocabulary cap for fairness and speed
min_df       <- 10         # drop very rare terms: min docs containing term
max_df_prop  <- 0.5        # drop very common terms: max doc proportion
random_state <- 0          # reproducibility
```

Data

Load the Associated Press corpus

`topicmodels` ships a preprocessed **DocumentTermMatrix** named `AssociatedPress` (AP).

```r
data("AssociatedPress", package = "topicmodels")
dtm <- AssociatedPress  # class: DocumentTermMatrix (tm + slam)
n_docs <- nrow(dtm); n_terms <- ncol(dtm)
n_docs; n_terms
```

```
## [1] 2246
```

```
## [1] 10473
```

Shared Vocabulary & Matrices (Fair Comparison)

We build **one** filtered vocabulary and use it for both models:

- **Counts (DTM)** → input to **LDA**
- **TF–IDF** (computed from those same counts) → input to **NMF**

Filtering steps:

- Remove **rare** terms (`min_df`) and **overly common** terms (`max_df_prop`).
- Cap to the `max_features` most frequent terms.
- Keep everything sparse.

```
# Document frequency: number of documents containing the term
df <- slam::col_sums(dtm > 0)
keep <- (df >= min_df) & (df <= max_df_prop * n_docs)

# Limit to top features by overall term frequency (within 'keep')
tf  <- slam::col_sums(dtm[, keep])
ord <- order(tf, decreasing = TRUE)
if (length(ord) > max_features) ord <- ord[seq_len(max_features)]
keep_terms <- names(tf)[ord]

# Apply selection
dtm_filt <- dtm[, keep_terms]
dtm_filt <- dtm_filt[, slam::col_sums(dtm_filt) > 0]  # ensure non-empty columns

# Drop empty documents
nz_docs <- slam::row_sums(dtm_filt) > 0
dtm_filt <- dtm_filt[nz_docs, ]

# Refresh doc count for downstream code
n_docs <- nrow(dtm_filt)

dim(dtm_filt)
```

```
## [1] 2245 5000
```

Build TF–IDF (sublinear TF + L2 row norm)

We approximate scikit-learn's default TF–IDF:

- **TF:** sublinear scaling `log1p(count)`
- **IDF:** `log((1 + N) / (1 + df)) + 1`
- **Normalize:** L2 per document (row)

Sublinear scaling means replacing raw term frequency with a concave transform—typically $\mathbf{log(1 + tf)}$ (or sometimes $\sqrt{\text{tf}}$). It gives **diminishing returns** for repeated occurrences of the same word, so a term appearing 50 times isn't treated 50× more important than one appearing 5 times. This **dampens bursty/repetitive words**, reduces bias toward longer documents, and generally yields more stable, interpretable weights in TF-IDF, retrieval, and topic modeling.

```
# Convert DTM (slam::simple_triplet_matrix) to a sparse dgCMatrix (Matrix package)
stm <- dtm_filt

# Use existing document/term names if present; otherwise create synthetic IDs.
doc_ids  <- rownames(stm); if (is.null(doc_ids))  doc_ids  <- paste0("doc_",  seq_len(stm$nrow))
term_ids <- colnames(stm); if (is.null(term_ids)) term_ids <- paste0("term_", seq_len(stm$ncol))

# Build a column-compressed sparse matrix (dgCMatrix) from slam triplet slots:
#   stm$i -> row indices (docs)
```

```r
#   stm$j -> column indices (terms)
#   stm$v -> counts (term frequencies)
# Provide dims and human-friendly dimnames for later inspection.
V_counts <- sparseMatrix(
  i = stm$i, j = stm$j, x = stm$v,
  dims = c(stm$nrow, stm$ncol),
  dimnames = list(Docs = doc_ids, Terms = term_ids)
)

# --- TF-IDF (sublinear TF + IDF + L2 row norm) ---

# Sublinear TF: replace raw counts tf with log(1 + tf).
# Access nonzero values directly via the @x slot for efficiency.
V_tf <- V_counts
V_tf@x <- log1p(V_tf@x)

# IDF on the filtered DTM:
# df_filt = #docs containing each term (document frequency).
df_filt <- slam::col_sums(dtm_filt > 0)

# Use the filtered doc count N. If you prefer this chunk self-contained,
# uncomment the next line and remove earlier n_docs definitions:
# n_docs <- nrow(stm)

# Smoothed IDF to avoid div-by-zero and overly extreme values:
# idf_j = log((1 + N) / (1 + df_j)) + 1
idf <- log((1 + n_docs) / (1 + as.numeric(df_filt))) + 1

# Apply IDF on the right: (docs x terms) %*% diag(idf) scales each term column by its IDF.
V_tfidf <- V_tf %*% Diagonal(x = idf)

# L2-normalize each document vector to unit length:
# row_norm_d = sqrt(sum_t (tfidf_{d,t}^2)).
row_norm <- sqrt(rowSums(V_tfidf^2))

# Guard against zero rows so scaling is safe.
row_norm[row_norm == 0] <- 1

# Left-multiply by diag(1/row_norm) to scale each row to unit L2 norm.
V_tfidf <- Diagonal(x = 1 / row_norm) %*% V_tfidf

# Report final matrix shape: (#docs, #terms)
dim(V_tfidf)
```

```
## [1] 2245 5000
```

Fit the Models

LDA on counts (variational EM)

VEM (Variational EM) for LDA is a deterministic inference method that alternates between a **variational E-step** (estimate per-document topic proportions and per-word topic assignments by maximizing the

ELBO) and an **M-step** (update global topic–word distributions and sometimes hyperparameters). It's fast, stable given a seed, and works directly with **count** DTMs.

```
control_lda <- list(estimate.alpha = TRUE, seed = random_state)
lda_model <- LDA(dtm_filt, k = n_topics, method = "VEM", control = control_lda)
lda_model
```

```
## A LDA_VEM topic model with 10 topics.
```

NMF on counts (KL / 'brunet')

We use KL divergence via the Brunet multiplicative updates, initialized with NNDSVD, on the **count** matrix. This setting typically yields more interpretable topics on news corpora than Euclidean on TF–IDF, even with a single run.

The KL-divergence–based **multiplicative updates** algorithm (Brunet et al., 2004) that minimizes $D_{\mathrm{KL}}(V|WH)$. It's well-suited to **count-like** data, tends to produce sparse, parts-based topics, and benefits from good initialization (e.g., **NNDSVD**) or a few replicates to avoid poor local minima.

```
set.seed(random_state)
nmf_model <- suppressWarnings(
  nmf(as.matrix(V_counts), rank = n_topics, method = "brunet",
      nrun = 1, seed = "nndsvd")
)
nmf_model
```

```
## <Object of class: NMFfit>
##  # Model:
##   <Object of class:NMFstd>
##   features: 2245
##   basis/rank: 10
##   samples: 5000
##  # Details:
##   algorithm:  brunet
##   seed:  nndsvd
##   RNG: 10403L, 624L, ..., -2085092265L [bb41ddce3f896a749dea08ac0f513ce0]
##   distance metric:  'KL'
##   residuals:  991481.4
##   Iterations: 2000
##   Timing:
##      user  system elapsed
##   697.456  26.053 726.213
```

Topic Inspection

Extract **top words** for each topic.

```
# LDA: topic-word probabilities (phi), dimensions: k x vocab
lda_post <- posterior(lda_model)
phi_lda  <- lda_post$terms

# NMF: basis matrix W (features x k); convert to topic-word (k x features) and row-normalize
W <- basis(nmf_model)
phi_nmf <- t(W)
```

```r
phi_nmf <- phi_nmf / pmax(rowSums(phi_nmf), 1e-12)

feat_names <- colnames(as.matrix(dtm_filt))

top_words <- function(phi, topn = n_top_words, features = feat_names) {
  apply(phi, 1, function(row) {
    ids <- order(row, decreasing = TRUE)[seq_len(min(topn, length(row)))]
    paste(features[ids], collapse = " ")
  })
}

cat("### Topics in LDA (counts)\n")
```

```
## ### Topics in LDA (counts)
```

```r
tw_lda <- top_words(phi_lda, n_top_words, feat_names)
for (i in seq_along(tw_lda)) cat(sprintf("Topic #%d:\n%s\n\n", i-1, tw_lda[[i]]))
```

```
## Topic #0:
## company million new workers federal year corp president employees last inc pay billion plan departmen
##
## Topic #1:
## percent million year market stock billion prices new sales cents last rose higher rate trading index
##
## Topic #2:
## i years new people mrs two time year like just first dont school get family home yearold life think l
##
## Topic #3:
## soviet united union states gorbachev president west east government talks german germany communist mo
##
## Topic #4:
## air two flight plane officials navy space defense aircraft time force first military million three sp
##
## Topic #5:
## police government people two killed army south officials military city three reported arrested group
##
## Topic #6:
## oil iraq dollar united yen kuwait iraqi late gulf saudi iran states gold london war foreign president
##
## Topic #7:
## people state city water children new hospital health officials percent area two fire medical high aid
##
## Topic #8:
## bush president congress house party new bill percent budget government vote senate i committee year p
##
## Topic #9:
## court dukakis i state attorney case bush drug trial campaign president judge jackson told federal cha
```

```r
cat("### Topics in NMF (counts, KL)\n")
```

```
## ### Topics in NMF (counts, KL)
```

```r
tw_nmf <- top_words(phi_nmf, n_top_words, feat_names)
for (i in seq_along(tw_nmf)) cat(sprintf("Topic #%d:\n%s\n\n", i-1, tw_nmf[[i]]))
```

```
## Topic #0:
```

```
## missing continental delegates davis grant come camp flew constitutional kind session chosen committe
##
## Topic #1:
## told deadline burned amount relations victim labor sale run electronic republicans improve markets g
##
## Topic #2:
## hill sure temperatures rest growing assistant wanted overseas responded convention interview governo
##
## Topic #3:
## release forced organized game terms motor joint decades joined houses close remove peace internation
##
## Topic #4:
## marks reserve mr corn anniversary missile ruling court northern john movie art prison suggested prop
##
## Topic #5:
## studies granted prepared notice england just wine activists atlanta tell risk failing county accused
##
## Topic #6:
## doesnt hold residents chrysler fourth david fed chicago weather director h increased requirements gi
##
## Topic #7:
## full st list scheduled degree force talk contributions two immediately flying pressure process thurs
##
## Topic #8:
## connection corporate todays argued spending acquisition protection citizens worth materials downtown
##
## Topic #9:
## encourage ended boost columbia christian time steel refused class individual demonstrators iraqi new
```

Model-Agnostic Metrics

UMass Coherence

Question: Do a topic's top words tend to co-occur in the same documents?

We compute a log-based co-occurrence score across top-word pairs (less negative ≈ better).

```r
umass_coherence <- function(phi, dtm_counts, topn = 20, eps = 1e-12) {
  # Convert to sparse matrix (dgCMatrix)
  stm <- dtm_counts
  X <- sparseMatrix(i = stm$i, j = stm$j, x = stm$v,
                    dims = c(stm$nrow, stm$ncol))
  # Binarize
  X@x <- ifelse(X@x > 0, 1, 0)
  df <- Matrix::colSums(X)
  C  <- Matrix::t(X) %*% X

  # Top indices per topic
  top_ids <- apply(phi, 1, function(row)
    order(row, decreasing = TRUE)[seq_len(min(topn, length(row)))] )

  scores <- c()
  for (k in seq_len(nrow(phi))) {
```

```r
    ids <- top_ids[, k]
    s <- 0; pairs <- 0
    if (length(ids) >= 2) {
      for (i in 2:length(ids)) {
        wi <- ids[i]
        for (j in 1:(i-1)) {
          wj <- ids[j]
          co <- C[wi, wj]
          s <- s + log((as.numeric(co) + eps) / (as.numeric(df[wj]) + eps))
          pairs <- pairs + 1
        }
      }
    }
    scores <- c(scores, ifelse(pairs > 0, s / pairs, NA_real_))
  }
  mean(scores, na.rm = TRUE)
}
```

Mean Jensen–Shannon Distance (JSD) Between Topics

Question: How different are topics from each other overall?

Treat each topic as a probability distribution over terms; average the Jensen–Shannon divergence across topic pairs (higher = more distinct).

```r
row_normalize <- function(M, eps = 1e-12) {
  M / pmax(rowSums(M), eps)
}

kl_div <- function(p, q, eps = 1e-12) {
  p <- p + eps; q <- q + eps
  sum(p * log(p / q))
}

js_div <- function(p, q, eps = 1e-12) {
  m <- 0.5 * (p + q)
  0.5 * kl_div(p, m, eps) + 0.5 * kl_div(q, m, eps)
}

mean_js_distance <- function(phi) {
  phi <- row_normalize(phi)
  K <- nrow(phi)
  d <- c()
  for (i in 1:(K-1)) {
    for (j in (i+1):K) {
      d <- c(d, js_div(phi[i, ], phi[j, ]))
    }
  }
  mean(d)
}
```

Compute and Report Metrics

```r
lda_coh <- umass_coherence(phi_lda, dtm_filt, topn = n_top_words)
nmf_coh <- umass_coherence(phi_nmf, dtm_filt, topn = n_top_words)

lda_js  <- mean_js_distance(phi_lda)
nmf_js  <- mean_js_distance(phi_nmf)

cat("=== Summary Metrics ===\n")
```

```
## === Summary Metrics ===
```

```r
cat(sprintf("LDA  coherence (UMass): %.4f   | separation (JSD): %.4f\n", lda_coh, lda_js))
```

```
## LDA  coherence (UMass): -1.7940   | separation (JSD): 0.3788
```

```r
cat(sprintf("NMF  coherence (UMass): %.4f   | separation (JSD): %.4f\n", nmf_coh, nmf_js))
```

```
## NMF  coherence (UMass): -8.5110   | separation (JSD): 0.5543
```

Interpretation

- **UMass coherence**: Do a topic's top words tend to appear together in documents? (higher/less negative ≈ better).

- **Mean JSD**: How different are the topics from each other overall? (higher = more distinct).

- **Typical outcome**: LDA often has slightly better coherence; NMF (with narrower topics) often has higher separation.

- Maintain a **shared vocabulary** to keep comparisons fair.

UMass coherence (higher/less-negative is better):

- LDA: **-1.794** pretty reasonable on AP; LDA topics (markets, USSR, Gulf War, courts, etc.) are visibly coherent.
- NMF: **-8.5110** very low coherence; the top words read "spiky/odd" and don't co-occur much in documents. That's typical when NMF is run with **Euclidean loss on TF–IDF**: it emphasizes rare/idiosyncratic terms rather than co-occurring ones.

JSD separation (higher = topics more dissimilar as distributions):

- NMF: **0.554** > LDA: **0.379**. NMF topics can become very "disjoint" (few shared high-weight words), so they look **far apart** even if they're not coherent. High separation is not automatically good; coherence matters more for interpretability.