# 24 Embeddings for Scientometrics

*R. Heller · Date: 2026-05-09 · Last changed: 2026-10-31*

## 24.1 Learning objectives

After completing this chapter, you will be able to:

- Explain how word and document embeddings differ from bag-of-words representations
- Train simple word embeddings on a bibliometric corpus
- Use pre-trained embeddings to compute document similarity
- Apply UMAP or t-SNE to visualise document clusters in embedding space
- Assess when embeddings add value over simpler text representations

## 24.2 Setup

```
library(tidyverse)
library(openalexR)
library(quanteda)
library(word2vec)
library(uwot)
library(glue)
library(gt)

set.seed(20260509)

source(here::here("R", "api_helpers.R"))
source(here::here("R", "utils.R"))
source(here::here("R", "sci_palette.R"))
```

## 24.3 Conceptual background

Bag-of-words representations (21.3) treat each word as an independent dimension. “Citation analysis” and “bibliometric study” share no features despite being semantically related. **Word embeddings** address this by mapping words into a dense, low-dimensional vector space where semantically similar words are close together. The word2vec algorithm learns these representations by predicting words from their context (skip-gram) or context from words (CBOW) in a large training corpus.

**Document embeddings** extend the idea to whole documents. Simple approaches average the word vectors of a document’s words (mean pooling). More sophisticated approaches include Doc2Vec (Paragraph Vector), which learns document-specific vectors alongside word vectors, and transformer-based models like **SPECTER**, which is pre-trained on scientific paper titles and abstracts using citation-based training signals.

For scientometrics, embeddings enable several applications:

- **Semantic similarity**: Finding papers with similar meaning, even when they use different terminology.
- **Clustering**: Grouping papers by content in embedding space, complementing topic models (22.3).
- **Visualisation**: Projecting embeddings to 2D with UMAP or t-SNE to create interpretable "maps" of a corpus.
- **Anomaly detection**: Identifying papers that are semantically distant from their assigned field or cluster.

The trade-off is interpretability: while TF-IDF features are directly readable (the word "citation" has a specific weight), embedding dimensions are opaque. Embeddings are powerful for similarity and clustering but resist the kind of term-level interpretation that co-word analysis provides.

## 24.4 Worked example

### 24.4.1 Preparing text data

```
works ← fetch_openalex(
  entity = "works",
  primary_location.source.id = "S148561398",
  from_publication_date = "2019-01-01",
  to_publication_date = "2023-12-31",
  options = list(sample = 400, seed = 42)
)

text_df ← works ▷
  filter(!is.na(abstract), nchar(abstract) > 100) ▷
  transmute(
    doc_id = id,
    title = display_name,
    text = paste(display_name, abstract, sep = ". "),
    year = year(publication_date),
    cited_by_count
  )

cat(glue("Documents for embedding: {nrow(text_df)}\n"))
```

```
#> Documents for embedding: 107
```

### 24.4.2 Training word2vec embeddings

```
clean_text ← text_df$text ▷
  str_to_lower() ▷
  str_replace_all("[^a-z ]", " ") ▷
  str_squish()

model ← word2vec(clean_text, dim = 100, iter = 20, min_count = 3,
                 type = "skip-gram", threads = 1)

cat(glue("Vocabulary size: {nrow(as.matrix(model))}\n"))
```

```
#> Vocabulary size: 1417
```

```
if ("citation" %in% rownames(as.matrix(model))) {
  nn <- predict(model, newdata = "citation", type = "nearest", top_n = 10)
  cat("Nearest neighbours of 'citation':\n")
  print(nn)
}
```

```
#> Nearest neighbours of 'citation':
#> $citation
#>       term1          term2 similarity rank
#> 1  citation     normalised      0.715    1
#> 2  citation        similar      0.655    2
#> 3  citation recommendation      0.653    3
#> 4  citation          mixed      0.653    4
#> 5  citation           cnci      0.644    5
#> 6  citation             pp      0.644    6
#> 7  citation  normalisation      0.643    7
#> 8  citation  bibliometrics      0.634    8
#> 9  citation           here      0.634    9
#> 10 citation         months      0.633   10
```

### 24.4.3 Computing document embeddings via mean pooling

Apply the same text preprocessing used for training: lowercase before removing characters outside the vocabulary alphabet. This preserves capitalised words and acronyms when matching them to the learned word vectors.

```
word_matrix <- as.matrix(model)

doc_embed <- function(text, wm) {
  words <- str_split(str_replace_all(str_to_lower(text), "[^a-z ]", " "),
                     "\\s+")[[1]]
  matched <- words[words %in% rownames(wm)]
  if (length(matched) == 0) return(rep(NA_real_, ncol(wm)))
  colMeans(wm[matched, , drop = FALSE])
}

doc_embeddings <- map(text_df$text, \(t) doc_embed(t, word_matrix))
embed_mat <- do.call(rbind, doc_embeddings)

valid <- complete.cases(embed_mat)
embed_mat <- embed_mat[valid, ]
text_df_valid <- text_df[valid, ]

cat(glue("Valid document embeddings: {nrow(embed_mat)}\n"))
```

```
#> Valid document embeddings: 107
```

### 24.4.4 Dimensionality reduction with UMAP

```
umap_result ← umap(embed_mat, n_neighbors = 15, min_dist = 0.1,
                   n_components = 2, ret_model = FALSE)

text_df_valid$umap_x ← umap_result[, 1]
text_df_valid$umap_y ← umap_result[, 2]
```

```
ggplot(text_df_valid, aes(x = umap_x, y = umap_y, colour = factor(year))) +
  geom_point(alpha = 0.6, size = 1.5) +
  scale_colour_manual(values = palette_sci(n_distinct(text_df_valid$year))) +
  labs(x = "UMAP 1", y = "UMAP 2", colour = "Year") +
  theme_sci() +
  theme(axis.text = element_blank(), axis.ticks = element_blank())
```

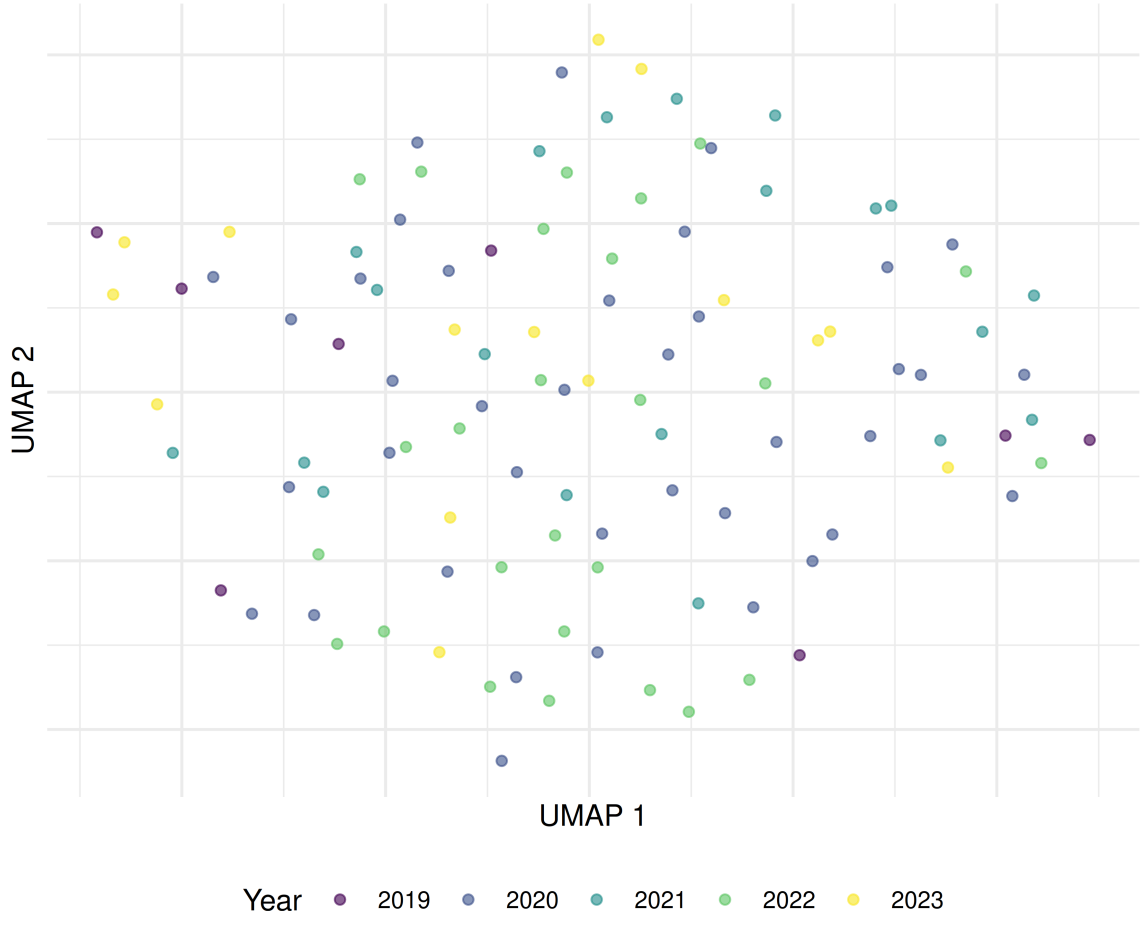



Figure 24.1: UMAP projection of document embeddings, coloured by publication year.

### 24.4.5 Document similarity search

```
cosine_sim ← function(a, b) sum(a * b) / (sqrt(sum(a^2)) * sqrt(sum(b^2)))

query_idx ← 1
sims ← map_dbl(seq_len(nrow(embed_mat)),
               \(i) cosine_sim(embed_mat[query_idx, ], embed_mat[i, ]))

top_similar ← order(sims, decreasing = TRUE)[2:6]

cat(glue("Query: {text_df_valid$title[query_idx]}\n\n"))
```

```
#> Query: The world-systemic network of global elite sociology: the western male
monoculture at faculties of the top one-hundred sociology departments of the world
```

```
cat("Most similar documents:\n")
```

```
#> Most similar documents:
```

```
walk(top_similar, \(i) {
  cat(glue("  [{round(sims[i], 3)}] {text_df_valid$title[i]}\n"))
})
```

```
#> [0.907] An analysis of international mobility and research productivity in computer
science[0.904] Publication outperformance among global South researchers: An analysis
of individual-level and publication-level predictors of positive deviance[0.894]
Coronavirus mapping in scientific publications: When science advances rapidly and
collectively, is access to this knowledge open to society?[0.891] Analysis of
scientific collaboration network of Italian Institute of Technology[0.887] Sample size
in bibliometric analysis
```

### 24.4.6 Embedding clusters vs. citation impact

```
ggplot(text_df_valid, aes(x = umap_x, y = umap_y,
                          colour = log1p(cited_by_count))) +
  geom_point(alpha = 0.6, size = 1.5) +
  scale_colour_viridis_c(option = "C", name = "log(cites + 1)") +
  labs(x = "UMAP 1", y = "UMAP 2") +
  theme_sci() +
  theme(axis.text = element_blank(), axis.ticks = element_blank())
```

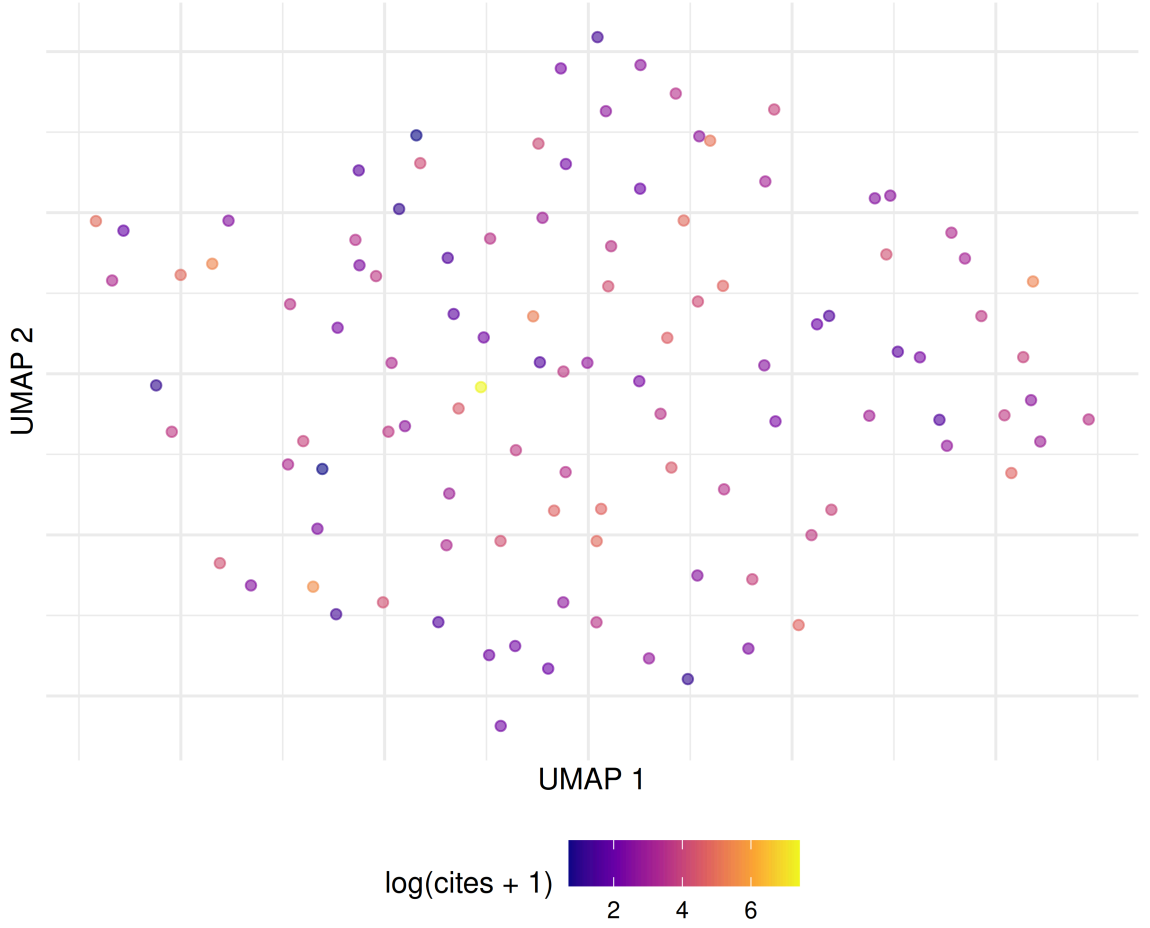



Figure 24.2: UMAP embedding coloured by citation impact (log scale).

## 24.5 Diagnostics and interpretation

- **Vocabulary coverage**: Check what fraction of abstract words appear in the embedding vocabulary. Low coverage (< 80%) indicates the training corpus is too small or preprocessing is too aggressive.
- **Nearest neighbours**: Inspect the nearest neighbours of key domain terms. If “citation” is closest to “references” and “bibliometric”, the embeddings capture domain structure.
- **UMAP stability**: UMAP is stochastic. Run with multiple seeds and check whether the global cluster structure is stable. Individual point positions will vary but clusters should persist.
- **Embedding dimensionality**: 100–300 dimensions is standard. Below 50, embeddings lose semantic resolution; above 300, training becomes slow with diminishing returns.

## 24.6 Limitations and responsible use

## 24.7 Limitations and responsible use

- **Embeddings are opaque.** Unlike TF-IDF weights, embedding dimensions have no interpretable meaning. You can measure similarity but not explain *why* two documents are similar without additional analysis.
- **Training data bias.** Embeddings reflect the biases of their training corpus. If the corpus overrepresents certain topics or perspectives, the embeddings will too.
- **Small corpora produce poor embeddings.** Word2vec needs tens of thousands of documents for reliable training. For small bibliometric corpora (< 1,000 papers), use pre-trained embeddings instead.
- **UMAP/t-SNE distort distances.** These methods preserve local structure but distort global distances. Documents that appear close in 2D may not be the most similar in high-dimensional space (Hicks et al. 2015).

## 24.8 Common pitfalls

## 24.9 Common pitfalls

- **Training on too few documents.** Word2vec on 200 abstracts produces unreliable embeddings. Use at least 5,000 documents or switch to pre-trained models.
- **Averaging without weighting.** Simple mean pooling treats every word equally. TF-IDF-weighted averaging gives more weight to distinctive terms and often produces better document embeddings.
- **Over-interpreting UMAP clusters.** Visual clusters in UMAP may reflect projection artifacts. Validate clusters with a separate method (e.g., k-means in the full embedding space).
- **Comparing embeddings from different models.** Embedding spaces are not aligned across different training runs. Cosine similarity is only meaningful within a single embedding space.

## 24.10 Exercises

1. **Weighted averaging.** Compute document embeddings using TF-IDF-weighted word vectors instead of simple means. Does the UMAP visualisation change?

2. **K-means clustering.** Apply k-means (k = 5, 8, 12) to the document embeddings. Compare the resulting clusters with LDA topics from 22.4. Do they agree?

3. **Pre-trained embeddings.** If available, use SPECTER embeddings (via reticulate and Python) instead of corpus-trained word2vec. How does similarity search quality change?

4. **Analogy test.** Test whether the word2vec model captures analogical relationships (e.g., "journal" - "article" + "book" ≈ "publisher"). What does the result tell you about the model?

## 24.11 Solutions

Solutions are provided in E.

## 24.12 Further reading

- Silge and Robinson (2017) — Text embeddings in R, including integration with word2vec.
- Priem et al. (2022) — OpenAlex concept embeddings and their relationship to document content.
- Waltman et al. (2010) — Bibliometric similarity measures as context for embedding-based approaches.

## 24.13 Session info

```
sessionInfo()
```

```
#> R version 4.6.1 (2026-06-24)
#> Platform: x86_64-pc-linux-gnu
#> Running under: Ubuntu 24.04.5 LTS
#>
#> Matrix products: default
#> BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
#> LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.26.so;  LAPACK
version 3.12.0
#>
#> locale:
#>  [1] LC_CTYPE=C.UTF-8       LC_NUMERIC=C           LC_TIME=C.UTF-8
#>  [4] LC_COLLATE=C.UTF-8     LC_MONETARY=C.UTF-8    LC_MESSAGES=C.UTF-8
#>  [7] LC_PAPER=C.UTF-8       LC_NAME=C              LC_ADDRESS=C
#> [10] LC_TELEPHONE=C         LC_MEASUREMENT=C.UTF-8 LC_IDENTIFICATION=C
#>
#> time zone: UTC
#> tzcode source: system (glibc)
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#>  [1] uwot_0.2.5                Matrix_1.7-5
#>  [3] word2vec_0.4.1            stm_1.3.10
#>  [5] topicmodels_0.2-17        quanteda.textstats_0.97.2
#>  [7] visNetwork_2.1.4          ggraph_2.2.2
#>  [9] tidygraph_1.3.1           igraph_2.3.4
#> [11] quanteda_4.5.1            pdftools_3.9.1
#> [13] arrow_26.0.1.1            bibliometrix_5.5.0
#> [15] RefManageR_1.4.0          bib2df_1.1.2.0
#> [17] rcrossref_1.2.1           gt_1.4.0
#> [19] tidytext_0.4.3            glue_1.8.1
#> [21] openalexR_3.1.2           lubridate_1.9.5
#> [23] forcats_1.0.1             stringr_1.6.0
#> [25] dplyr_1.2.1               purrr_1.2.2
#> [27] readr_2.2.0               tidyr_1.3.2
#> [29] tibble_3.3.1              ggplot2_4.0.4
#> [31] tidyverse_2.0.0
#>
#> loaded via a namespace (and not attached):
#>   [1] bibtex_0.5.3           RColorBrewer_1.1-3     rstudioapi_0.19.0
#>   [4] jsonlite_2.0.0         magrittr_2.0.5         modeltools_0.2-25
#>   [7] farver_2.1.2           rmarkdown_2.32         fs_2.1.0
#>  [10] vctrs_0.7.3            memoise_2.0.1          askpass_1.2.1
#>  [13] base64enc_0.1-6        htmltools_0.5.9        contentanalysis_1.1.1
#>  [16] curl_8.1.0             janeaustenr_1.0.0      cellranger_1.1.0
#>  [19] sass_0.5.0             bslib_0.12.0           htmlwidgets_1.6.4
#>  [22] tokenizers_0.3.0       plyr_1.8.9             httr2_1.3.0
#>  [25] plotly_4.12.1          cachem_1.1.0           dimensionsR_0.0.3
#>  [28] mime_0.13              lifecycle_1.0.5        pkgconfig_2.0.3
#>  [31] R6_2.6.1               fastmap_1.2.0          shiny_1.15.0
#>  [34] digest_0.6.39          patchwork_1.3.2        shinycssloaders_1.1.0
#>  [37] rprojroot_2.1.1        RSpectra_0.16-2        SnowballC_0.7.1
#>  [40] labeling_0.4.3         urltools_1.7.3.1       timechange_0.4.0
#>  [43] polyclip_1.10-7        httr_1.4.9             compiler_4.6.1
#>  [46] here_1.0.2             bit64_4.8.6            withr_3.0.3
#>  [49] S7_1.0.0               viridis_0.6.5          ggforce_0.5.0
#>  [52] MASS_7.3-65            bibliometrixData_0.3.0 tools_4.6.1
#>  [55] otel_0.3.0             stopwords_2.3          zip_3.0.2
#>  [58] httpuv_1.6.17          rentrez_1.2.4          promises_1.5.0
#>  [61] grid_4.6.1             stringdist_0.9.17      reshape2_1.4.5
#>  [64] generics_0.1.4         gtable_0.3.6           tzdb_0.5.0
#>  [67] rscopus_0.9.0          ca_0.72                data.table_1.18.6.1
```

```
#>  [70] hms_1.1.4              xml2_1.6.0             utf8_1.2.6
#>  [73] ggrepel_0.9.8          pillar_1.11.1          nsyllable_1.0.1
#>  [76] vroom_1.7.1            later_1.4.8            tweenr_2.0.3
#>  [79] lattice_0.22-9         FNN_1.1.4.1            bit_4.6.0
#>  [82] tidyselect_1.2.1       tm_0.7-20              miniUI_0.1.2
#>  [85] downlit_0.4.5          knitr_1.52             gridExtra_2.3.1
#>  [88] NLP_0.3-3              bookdown_0.48          stats4_4.6.1
#>  [91] crul_1.6.0             xfun_0.62              graphlayouts_1.2.5
#>  [94] matrixStats_1.5.0      DT_0.34.0              humaniformat_0.6.0
#>  [97] stringi_1.8.9          qpdf_1.4.1             yaml_2.3.12
#> [100] evaluate_1.0.5         httpcode_0.3.0         cli_3.6.6
#> [103] xtable_1.8-8           jquerylib_0.1.4        Rcpp_1.1.2
#> [106] readxl_1.6.0           triebeard_0.4.1        XML_3.99-0.25
#> [109] parallel_4.6.1         assertthat_0.2.1       pubmedR_1.1.0
#> [112] slam_0.1-56            viridisLite_0.4.3      scales_1.4.0
#> [115] crayon_1.5.3           openxlsx_4.2.9         rlang_1.3.0
#> [118] fastmatch_1.1-8
```