CSE2525 · DATA MINING · 2026–27

Wrap-up & Exam Prep

A fast recap of the three information-access topics — Embeddings · Graph Mining · Indexing Text — with the exact concepts, algorithms and skills that are examinable.

A. Anand · wrap-up study deck

THE MAP FOR THIS SESSION

Three topics, one way to revise each

Embeddings

- Distances vertical
- Dense, semantic features
- Word2vec & random walks

Graph Mining

- Matrices vertical
- Properties, centrality, Laplacian
- Spectral & Girvan–Newman

Indexing Text

- Counting vertical
- Inverted / forward indexes
- Query processing

For each topic revise the same three layers: Concepts (what & why) → Algorithms (how) → Skills (apply it). The "Exam Material" slides list exactly these.

PART I

Embeddings

Turning raw data types into dense, semantic vectors — unsupervised, automatic feature extraction.

HOW DO WE VECTORIZE OUR INPUT DATA?

The classic ML approach

THE PIPELINE

raw data (tabular · graph · text · …) → split into **train / test** → compute features → **learning algorithm** → **model** → **output**.

— Every learner needs a vector, but raw data types don't come as vectors.

— The hard question is the *first* step: how do we represent tabular, graph or text data as features?

— Classic answer — hand-crafted features. The modern answer is embeddings.

EMBEDDINGS = UNSUPERVISED AUTOMATIC FEATURE EXTRACTION

From hand-crafted to learned features

Hand-crafted features

- Sparse
- Large (high-dimensional)
- Non-semantic
- Built by hand

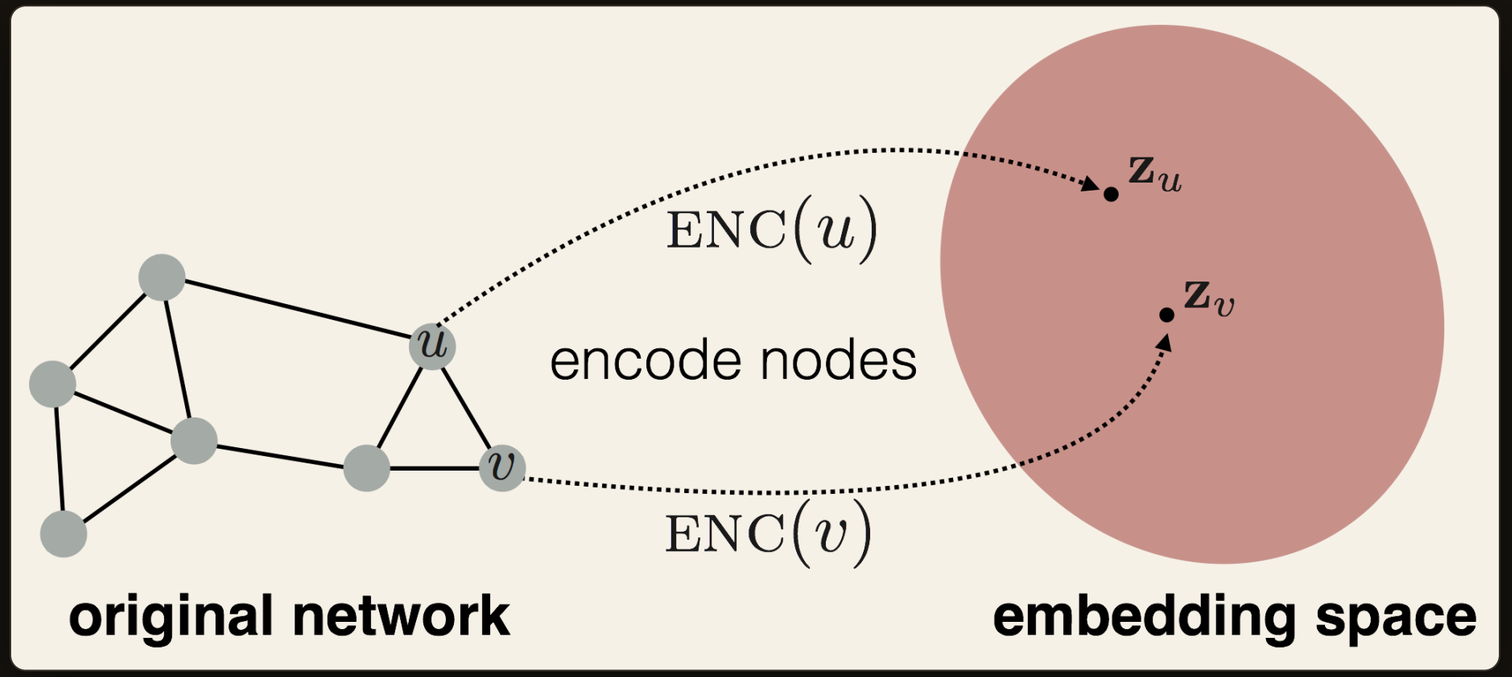
Embeddings

- Dense
- Smaller (low-dimensional)
- Semantic
- Extracted automatically

An embedding is learned without labels, then reused as the input features for downstream supervised tasks.

HIGH-DIMENSIONAL → LOW-DIMENSIONAL

Embeddings encode nodes (and words)



An encoder ENC(·) maps each node (or word) to a point so that similarity in the embedding space (e.g. a dot product $z_u \cdot z_v$) approximates similarity in the context. High-dimensional features → a few semantic ones.

EXAM MATERIAL · EMBEDDINGS

What to revise: Embeddings

Concepts

- Sparsity
- Dense representations
- Bag of words
- Random walks
- Embeddings table

Algorithms

- Word2vec
- Random-walk algorithms

Skills

- Applying embeddings to supervised tasks
- The role of context

PART II

Graph Mining

Graph properties, centrality and clustering — read through the lens of matrices and eigendecompositions.

GRAPHS AS MATRICES

Matrix representations & graph properties

FROM ADJACENCY MATRIX WE READ…

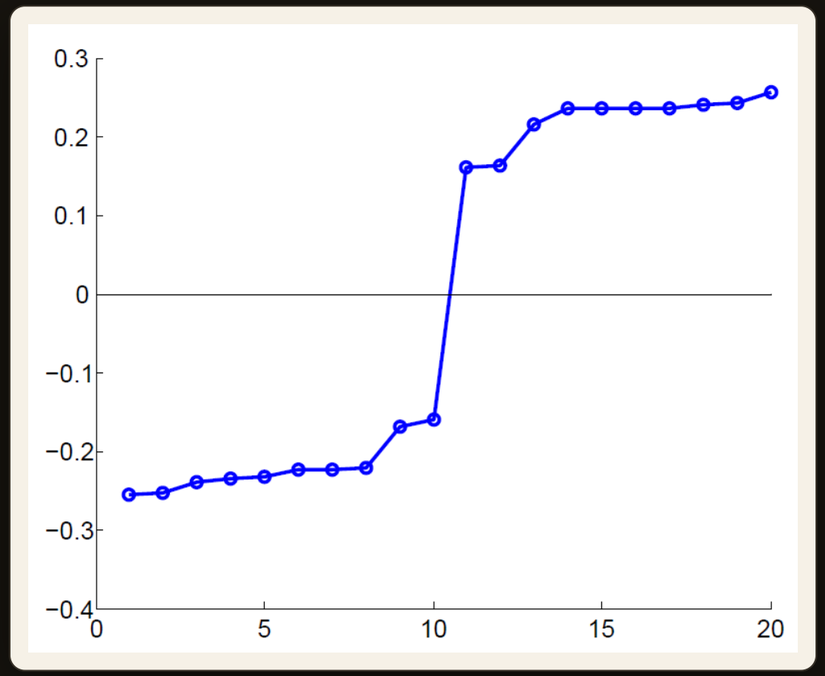
distance between nodes · **eccentricity** (farthest reachable) · **radius** (min eccentricity) · **diameter** (max eccentricity) · **centrality** measures.

The adjacency matrix A is symmetric for an undirected graph; row sums give node degree, and powers A^k count walks of length k.

	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

GRAPH PARTITIONING · THE FIEDLER VECTOR

Spectral clustering: split by eigenvector sign



The second-smallest eigenvector of the Laplacian (the Fiedler vector). Sorting its entries, the sign of each component assigns nodes to group A (negative) or B (positive) — a principled graph bisection.

GIRVAN–NEWMAN · CENTRALITY COMPUTATION

Community detection by edge betweenness

— Give each edge a credit = the credit that flows from parent → child in a breadth-first tree.

— Summed over all source nodes, this is the edge's betweenness — how many shortest paths run through it.

— Remove the highest-betweenness edge, recompute, and repeat; the graph falls apart into communities.

TWO ROUTES TO CLUSTERS

Spectral clustering cuts using the Laplacian's eigenvectors; Girvan–Newman cuts by repeatedly deleting high-credit edges. Both turn a graph into groups.

EXAM MATERIAL · GRAPH MINING

What to revise: Graph Mining

Concepts

- Graph representation
- Properties based on distances
- Centrality measures
- Laplacian
- Eigendecompositions

Algorithms

- Spectral clustering
- Girvan–Newman algorithm

Skills

- Applying spectral decomposition to clustering
- Generalizing bisection to multiple clusters

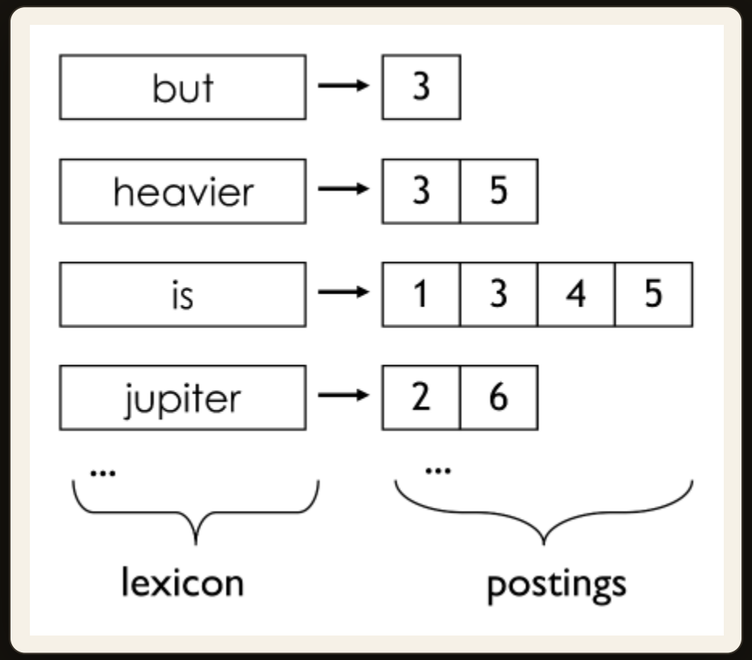
PART III

Indexing Text

The data structures that let us search massive text collections fast — the inverted index and its dictionary.

THE CORE STRUCTURE

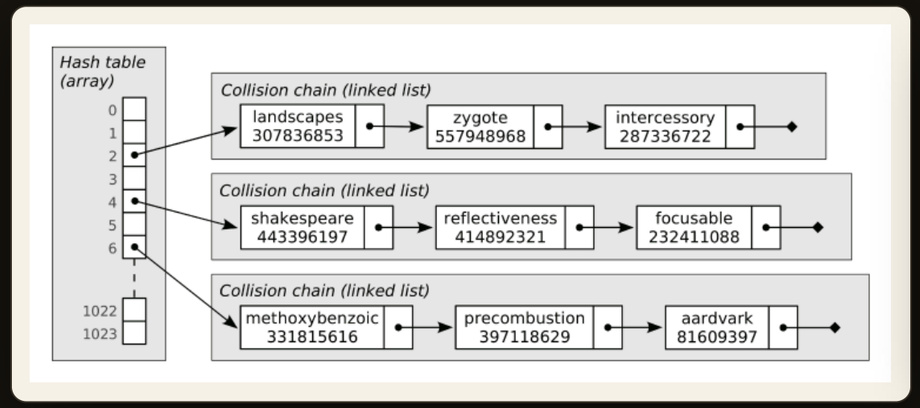
The inverted index

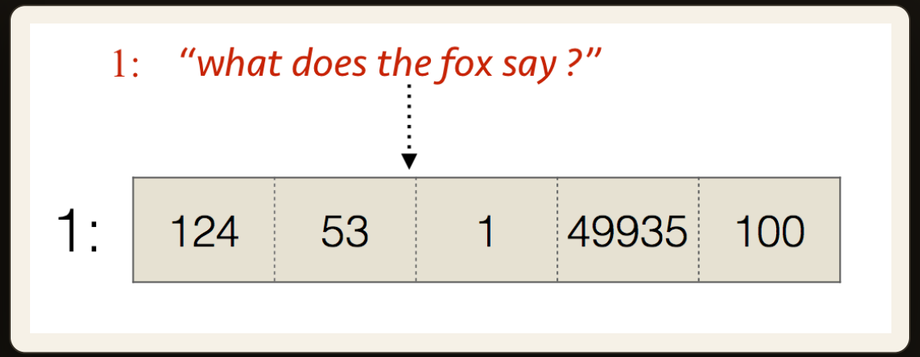


A lexicon (dictionary) of terms, each pointing to a postings list of the document IDs that contain it — e.g. is → 1, 3, 4, 5. Answering a query = intersecting / merging postings lists.

DATA STRUCTURES FOR INDEXING

Dictionary lookup & the forward index





Left: the dictionary as a hash table with collision chains, mapping a term to its ID. Right: a forward index — each document stored as its sequence of term IDs.

EXAM MATERIAL · INDEXING TEXT

What to revise: Indexing Text

Concepts

- Inverted indexes
- Dictionary
- Forward indexes
- MapReduce

Algorithms

- Query processing algorithms
- Indexing algorithms

Skills

- Dealing with large text collections
- Dealing with query semantics

THE SPINE OF THE WHOLE COURSE

Everything hangs off three verticals

Distances

- Similarity & metrics
- **Embeddings** for words & nodes
- Context-based similarity

Matrices

- Adjacency & Laplacian
- Eigendecomposition / PCA
- **Graph mining** & clustering

Counting

- Hashing & sketching
- Dictionaries & postings
- **Indexing text** at scale

When a new exam question appears, first ask: *is this fundamentally a distance, a matrix, or a counting problem?*

HOW TO PREPARE

Exam preparation checklist

FOR EACH TOPIC, BE ABLE TO…

State the concepts and *why* they matter, trace the algorithms by hand on a tiny example, and apply the skill to an unseen problem.

Worked examples worth re-running: encode nodes with Word2vec / random walks, bisect a graph with the Fiedler vector, delete edges with Girvan–Newman, and answer a query over an inverted index.

- Closed book, calculator allowed · open and multiple-choice questions · no programming this year.
- Only the selected chapters on Brightspace are examinable; the homework sets mirror exam type and difficulty.

WRAP-UP · ONE SLIDE

Summary

TAKE AWAY

Embeddings turn raw data into dense, semantic vectors (unsupervised); graph mining reads structure through matrices, the Laplacian and eigendecompositions; indexing makes massive text searchable with inverted and forward indexes.

All three sit on the course spine — Distances · Matrices · Counting. Revise each as Concepts → Algorithms → Skills, and practise the worked examples by hand. Good luck on the exam.