COMP 019 - Applications Development and Emerging Technologies

Emerging Technologies: MCP, A2A, Pgvector

SESSION 11: VECTOR DATABASES & PGVECTOR (EMERGING TECHNOLOGY)

ACTIVITY 11.1: INTRODUCTION TO VECTOR DATABASES

Topic: Understanding embeddings and vector similarity search

Description: Learn the fundamentals of vector databases for AI applications

INSTRUCTIONS:

Understand vector concepts:
- What are embeddings?
- Text → Vector representation
- Semantic similarity via vector distance
- Use cases: search, recommendations, RAG

Vector database options:
- Pgvector (PostgreSQL extension)
- Pinecone, Weaviate, Chroma (dedicated)
- Why Pgvector: Use existing PostgreSQL!

Embedding models:
- OpenAI text-embedding-ada-002
- Sentence Transformers (free, local)
- Google's embedding models

How semantic search works:
- Store text with embedding vectors
- Query: Convert question to vector
- Find similar vectors in database
- Return matching content

Deliverables: Submit vector database concepts document with diagrams	**25 Points**

ACTIVITY 11.2: SETTING UP PGVECTOR

Topic: Adding vector capabilities to PostgreSQL

Description: Install and configure Pgvector extension for vector search

INSTRUCTIONS:

Enable Pgvector:

Cloud PostgreSQL (Supabase/Neon):
- Usually pre-installed, just enable extension
- CREATE EXTENSION vector;

Local PostgreSQL:
• Install pgvector extension
• Enable in database
Create vector-enabled table:
• Add embedding column: vector(1536)
• Store: id, content, embedding
• Create index for fast search
Django integration:
• pip install pgvector
• Add VectorField to model
• Migration to add vector column
Generate embeddings:
• Use sentence-transformers or API
• pip install sentence-transformers
• Generate embedding for sample texts
• Store in database

Deliverables: Submit Pgvector setup code and Django model with vectors	**35 Points**

ACTIVITY 11.3: BUILDING RAG (RETRIEVAL-AUGMENTED GENERATION)

Topic: Creating AI-powered search with vector retrieval

Description: Build a RAG system combining vector search with AI generation

INSTRUCTIONS:

Understand RAG architecture:
• User asks question
• Convert question to embedding
• Search vector database for relevant content
• Send content + question to LLM
• LLM generates answer using retrieved content
Implement RAG system:
• Create embeddings for your content (student records, course materials)
• Store in Pgvector-enabled PostgreSQL
• Build search function
• Integrate with AI (Claude API or local model)
Build 'Smart Campus Assistant':
• Index: Course descriptions, policies, FAQs
• User asks: 'What are the requirements for the IT program?'
• System retrieves relevant documents
• AI generates helpful, accurate answer
Test with various queries
Compare: With RAG vs Without RAG

Deliverables: Submit RAG implementation code and comparison demo	**40 Points**

TOTAL POINTS: 100