"""

Configuration settings for the RAG pipeline.

This module contains all configuration parameters used throughout the application,

including model settings, chunk parameters, and API configurations.

"""

import os

from pathlib import Path

from typing import Dict, Any

from pydantic import BaseSettings, Field, validator

class Settings(BaseSettings):

"""Application settings with validation and environment variable support."""

# API Keys

openai\_api\_key: str = Field(..., env="OPENAI\_API\_KEY")

langsmith\_api\_key: str = Field(None, env="LANGSMITH\_API\_KEY")

# Model Configuration

llm\_model: str = Field("gpt-3.5-turbo", env="LLM\_MODEL")

embedding\_model: str = Field("text-embedding-ada-002", env="EMBEDDING\_MODEL")

# Model Parameters

temperature: float = Field(0.7, ge=0.0, le=2.0)

max\_tokens: int = Field(500, ge=1, le=4000)

top\_p: float = Field(1.0, ge=0.0, le=1.0)

frequency\_penalty: float = Field(0.0, ge=-2.0, le=2.0)

presence\_penalty: float = Field(0.0, ge=-2.0, le=2.0)

# Chunking Configuration

chunk\_size: int = Field(1000, ge=100, le=4000)

chunk\_overlap: int = Field(200, ge=0, le=500)

# Vector Store Configuration

vector\_store\_persist\_directory: str = Field("./data/vector\_store")

collection\_name: str = Field("rag\_documents")

# Retrieval Configuration

retriever\_search\_type: str = Field("similarity")

retriever\_search\_kwargs: Dict[str, Any] = Field(default\_factory=lambda: {"k": 4})

# Logging Configuration

log\_level: str = Field("INFO", env="LOG\_LEVEL")

log\_file: str = Field("./logs/rag\_pipeline.log")

# Data Paths

data\_directory: Path = Field(Path("./data"))

documents\_directory: Path = Field(Path("./data/documents"))

evaluation\_directory: Path = Field(Path("./data/evaluation"))

# Application Configuration

environment: str = Field("development", env="ENVIRONMENT")

debug: bool = Field(False, env="DEBUG")

@validator("data\_directory", "documents\_directory", "evaluation\_directory", pre=True)

def create\_directories(cls, v):

"""Ensure directories exist."""

path = Path(v)

path.mkdir(parents=True, exist\_ok=True)

return path

@validator("vector\_store\_persist\_directory", pre=True)

def create\_vector\_store\_directory(cls, v):

"""Ensure vector store directory exists."""

path = Path(v)

path.mkdir(parents=True, exist\_ok=True)

return str(path)

class Config:

"""Pydantic configuration."""

env\_file = ".env"

env\_file\_encoding = "utf-8"

case\_sensitive = False

# Create global settings instance

settings = Settings()