Table of Contents

# C. CONFIGURACIÓN DE AMBIENTE

## C.1 Introducción

Este anexo proporciona las instrucciones detalladas para configurar el ambiente de desarrollo y ejecución del sistema RAG para recuperación semántica de documentación técnica de Microsoft Azure. La configuración se basa en las dependencias especificadas en requirements.txt y ha sido validada en los entornos utilizados durante la investigación experimental.

## C.2 Requisitos del Sistema

### C.2.1 Requisitos Mínimos de Hardware

**Para Desarrollo y Testing:** - **CPU**: 4 cores mínimo (Intel i5/AMD Ryzen 5 o superior) - **RAM**: 8GB mínimo, 16GB recomendado - **Almacenamiento**: 10GB espacio libre (SSD recomendado) - **Red**: Conexión a internet estable para APIs y descargas de modelos

**Para Producción (Corpus completo):** - **CPU**: 8+ cores (Intel i7/Xeon o AMD Ryzen 7+) - **RAM**: 32GB mínimo para ChromaDB con 800K+ vectores - **Almacenamiento**: 50GB+ SSD (ChromaDB + modelos + datos) - **GPU**: Opcional, mejora rendimiento de embeddings (CUDA compatible)

### C.2.2 Requisitos de Software

* **Sistema Operativo**: Linux (Ubuntu 20.04+), macOS (10.15+), Windows 10+
* **Python**: 3.8.0 o superior (3.9.x recomendado)
* **pip**: Última versión
* **Git**: Para clonado del repositorio

## C.3 Configuración del Ambiente Python

### C.3.1 Creación de Ambiente Virtual

# Crear ambiente virtual  
python -m venv venv\_support\_model  
  
# Activar ambiente (Linux/macOS)  
source venv\_support\_model/bin/activate  
  
# Activar ambiente (Windows)  
venv\_support\_model\Scripts\activate

### C.3.2 Instalación de Dependencias

#### C.3.2.1 Instalación Estándar

# Actualizar pip  
pip install --upgrade pip  
  
# Instalar dependencias desde requirements.txt  
pip install -r requirements.txt

#### C.3.2.2 Dependencias Principales (requirements.txt)

# Core APIs y Base de Datos  
openai==1.93.0 # API de OpenAI para Ada embeddings y evaluación  
chromadb==0.5.23 # Base de datos vectorial principal  
python-dotenv==1.1.1 # Gestión de variables de ambiente  
  
# Interfaz de Usuario y Visualización  
streamlit==1.46.1 # Aplicación web interactiva  
plotly==6.2.0 # Gráficos interactivos  
weasyprint==63.1 # Generación de reportes PDF  
kaleido==0.2.1 # Backend para exportación de gráficos  
  
# Machine Learning y NLP  
scikit-learn==1.7.0 # Métricas de evaluación y utilidades ML  
torch==2.2.2 # Backend para modelos de transformers  
transformers==4.44.0 # Modelos de lenguaje y CrossEncoders  
sentence-transformers==5.0.0 # Modelos de embedding especializados  
accelerate==0.32.1 # Optimización de inferencia  
bitsandbytes==0.43.0 # Cuantización de modelos  
  
# Computación Científica  
numpy==1.26.4 # Operaciones numéricas fundamentales  
  
# Evaluación y Métricas  
bert-score==0.3.13 # Evaluación semántica con BERT  
rouge-score==0.1.2 # Métricas de evaluación de texto  
  
# APIs de Google (para modelos alternativos)  
google-generativeai==0.8.5 # API de Google Gemini  
google-auth==2.40.3 # Autenticación Google  
google-auth-oauthlib==1.2.2 # OAuth para Google APIs  
google-api-python-client==2.175.0 # Cliente Python para APIs Google

### C.3.3 Verificación de Instalación

# Verificar instalación Python  
python --version  
  
# Verificar instalación de paquetes críticos  
python -c "import openai; print('OpenAI:', openai.\_\_version\_\_)"  
python -c "import chromadb; print('ChromaDB:', chromadb.\_\_version\_\_)"  
python -c "import sentence\_transformers; print('Sentence-Transformers:', sentence\_transformers.\_\_version\_\_)"  
python -c "import streamlit; print('Streamlit:', streamlit.\_\_version\_\_)"

## C.4 Configuración de Variables de Ambiente

### C.4.1 Archivo .env

Crear archivo .env en el directorio raíz del proyecto:

# Archivo .env - NO INCLUIR EN CONTROL DE VERSIONES  
  
# API Keys (requeridas)  
OPENAI\_API\_KEY=your\_openai\_api\_key\_here  
  
# APIs opcionales (para funcionalidades extendidas)  
GOOGLE\_API\_KEY=your\_google\_api\_key\_here  
  
# Configuración de ChromaDB  
CHROMADB\_PATH=/Users/haroldgomez/chromadb2  
CHROMADB\_HOST=localhost  
CHROMADB\_PORT=8000  
  
# Configuración de modelos  
DEFAULT\_EMBEDDING\_MODEL=multi-qa-mpnet-base-dot-v1  
RERANKER\_MODEL=ms-marco-MiniLM-L-6-v2  
  
# Configuración de evaluación  
EVAL\_TOP\_K=10  
EVAL\_BATCH\_SIZE=4

### C.4.2 Variables de Sistema (Opcional)

# Configurar variables permanentes (Linux/macOS)  
echo 'export OPENAI\_API\_KEY="your\_key\_here"' >> ~/.bashrc  
echo 'export CHROMADB\_PATH="/path/to/chromadb"' >> ~/.bashrc  
source ~/.bashrc  
  
# Windows (PowerShell)  
[Environment]::SetEnvironmentVariable("OPENAI\_API\_KEY", "your\_key\_here", "User")

## C.5 Configuración de ChromaDB

### C.5.1 Inicialización de Base de Datos

# Script de inicialización (initialize\_chromadb.py)  
import chromadb  
from chromadb.config import Settings  
  
# Crear cliente ChromaDB  
client = chromadb.PersistentClient(  
 path="/Users/haroldgomez/chromadb2",  
 settings=Settings(  
 anonymized\_telemetry=False,  
 allow\_reset=True  
 )  
)  
  
print("ChromaDB inicializado correctamente")  
print(f"Ubicación: /Users/haroldgomez/chromadb2")

### C.5.2 Verificación de Colecciones

# Verificar estado de colecciones  
def verify\_collections():  
 collections = client.list\_collections()  
 print(f"Total colecciones: {len(collections)}")  
   
 expected\_collections = [  
 'docs\_ada', 'docs\_mpnet', 'docs\_minilm', 'docs\_e5large',  
 'questions\_ada', 'questions\_mpnet', 'questions\_minilm',   
 'questions\_e5large', 'questions\_withlinks'  
 ]  
   
 for name in expected\_collections:  
 try:  
 collection = client.get\_collection(name)  
 count = collection.count()  
 print(f"✅ {name}: {count:,} elementos")  
 except Exception as e:  
 print(f"❌ {name}: No encontrada - {e}")  
  
verify\_collections()

## C.6 Configuración de Modelos de Embedding

### C.6.1 Descarga Automática de Modelos

Los modelos se descargan automáticamente en el primer uso:

# Test de modelos  
from sentence\_transformers import SentenceTransformer  
  
# Modelos utilizados en el proyecto  
models = [  
 'all-MiniLM-L6-v2', # MiniLM - 384D  
 'multi-qa-mpnet-base-dot-v1', # MPNet - 768D   
 'intfloat/e5-large-v2' # E5-Large - 1024D  
]  
  
for model\_name in models:  
 try:  
 model = SentenceTransformer(model\_name)  
 print(f"✅ {model\_name}: Cargado correctamente")  
   
 # Test de embedding  
 test\_text = "Azure Virtual Machine configuration"  
 embedding = model.encode([test\_text])  
 print(f" Dimensiones: {embedding.shape}")  
   
 except Exception as e:  
 print(f"❌ {model\_name}: Error - {e}")

### C.6.2 Configuración de CrossEncoder

# Test de CrossEncoder para reranking  
from sentence\_transformers import CrossEncoder  
  
reranker = CrossEncoder('ms-marco-MiniLM-L-6-v2')  
print("✅ CrossEncoder cargado correctamente")  
  
# Test de reranking  
query = "How to configure Azure storage?"  
documents = [  
 "Azure Storage configuration guide",  
 "Virtual machine setup instructions"  
]  
  
scores = reranker.predict([(query, doc) for doc in documents])  
print(f" Scores de ejemplo: {scores}")

## C.7 Configuración de la Aplicación Streamlit

### C.7.1 Configuración Básica

Crear archivo .streamlit/config.toml en el directorio raíz del proyecto:

[global]  
dataFrameSerialization = "legacy"  
  
[server]  
port = 8501  
address = "localhost"  
maxUploadSize = 200  
  
[browser]  
gatherUsageStats = false  
  
[theme]  
primaryColor = "#0078d4" # Azure blue  
backgroundColor = "#ffffff"  
secondaryBackgroundColor = "#f5f5f5"  
textColor = "#000000"

### C.7.2 Test de Aplicación

# Ejecutar aplicación Streamlit principal  
streamlit run src/apps/main\_qa\_app.py  
  
# O ejecutar página de resultados experimentales  
streamlit run src/apps/cumulative\_metrics\_results\_matplotlib.py  
  
# Debería abrir automáticamente en: http://localhost:8501

## C.8 Resolución de Problemas Comunes

### C.8.1 Errores de Instalación

**Error: “Microsoft Visual C++ 14.0 is required” (Windows)**

# Solución: Instalar Visual Studio Build Tools  
# Descargar desde: https://visualstudio.microsoft.com/downloads/

**Error: “Failed building wheel for [package]”**

# Solución: Actualizar pip y setuptools  
pip install --upgrade pip setuptools wheel  
pip install --no-cache-dir -r requirements.txt

### C.8.2 Errores de Memoria

**Error: “RuntimeError: [enforce fail at CPUAllocator.cpp]”**

# Solución: Reducir batch size en evaluaciones  
export OMP\_NUM\_THREADS=1  
export MKL\_NUM\_THREADS=1

### C.8.3 Errores de ChromaDB

**Error: “ConnectionError: Could not connect to ChromaDB”**

# Solución: Verificar permisos y path  
import os  
chromadb\_path = "/Users/haroldgomez/chromadb2"  
os.makedirs(chromadb\_path, exist\_ok=True)  
os.chmod(chromadb\_path, 0o755)

### C.8.4 Errores de API

**Error: “OpenAI API rate limit exceeded”**

# Solución: Implementar rate limiting  
import time  
from openai import RateLimitError  
  
def safe\_api\_call(func, \*args, \*\*kwargs):  
 max\_retries = 3  
 for attempt in range(max\_retries):  
 try:  
 return func(\*args, \*\*kwargs)  
 except RateLimitError:  
 if attempt < max\_retries - 1:  
 time.sleep(2 \*\* attempt) # Exponential backoff  
 else:  
 raise

## C.9 Configuración para Desarrollo

### C.9.1 Herramientas de Desarrollo

# Instalación de herramientas de desarrollo  
pip install pytest black flake8 jupyter  
  
# Formateo de código  
black src/ Docs/Analisis/  
  
# Linting  
flake8 src/ Docs/Analisis/ --max-line-length=88

### C.9.2 Pre-commit Hooks (Opcional)

# Instalación de pre-commit  
pip install pre-commit  
  
# Crear .pre-commit-config.yaml  
cat > .pre-commit-config.yaml << EOF  
repos:  
- repo: https://github.com/psf/black  
 rev: 23.1.0  
 hooks:  
 - id: black  
- repo: https://github.com/pycqa/flake8  
 rev: 6.0.0  
 hooks:  
 - id: flake8  
 args: [--max-line-length=88]  
EOF  
  
# Instalar hooks  
pre-commit install

## C.10 Configuración para Producción

### C.10.1 Optimizaciones de Performance

# Variables de ambiente para producción  
export TOKENIZERS\_PARALLELISM=false  
export OMP\_NUM\_THREADS=4  
export MKL\_NUM\_THREADS=4  
export CUDA\_VISIBLE\_DEVICES=0 # Si GPU disponible

### C.10.2 Configuración de Logging

# logging\_config.py  
import logging  
  
def setup\_logging():  
 logging.basicConfig(  
 level=logging.INFO,  
 format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',  
 handlers=[  
 logging.FileHandler('support\_model.log'),  
 logging.StreamHandler()  
 ]  
 )  
  
setup\_logging()

## C.11 Verificación Final del Ambiente

### C.11.1 Script de Verificación Completa

# verify\_environment.py  
import sys  
import subprocess  
import importlib  
  
def verify\_environment():  
 print("🔍 VERIFICACIÓN DEL AMBIENTE")  
 print("=" \* 50)  
   
 # Verificar Python  
 print(f"Python: {sys.version}")  
   
 # Verificar dependencias críticas  
 critical\_packages = [  
 'openai', 'chromadb', 'streamlit', 'sentence\_transformers',  
 'torch', 'transformers', 'numpy', 'scikit-learn'  
 ]  
   
 for package in critical\_packages:  
 try:  
 module = importlib.import\_module(package)  
 version = getattr(module, '\_\_version\_\_', 'Unknown')  
 print(f"✅ {package}: {version}")  
 except ImportError:  
 print(f"❌ {package}: No instalado")  
   
 # Verificar APIs  
 import os  
 if os.getenv('OPENAI\_API\_KEY'):  
 print("✅ OpenAI API Key: Configurada")  
 else:  
 print("⚠️ OpenAI API Key: No configurada")  
   
 # Verificar ChromaDB  
 try:  
 import chromadb  
 client = chromadb.PersistentClient(path="/Users/haroldgomez/chromadb2")  
 collections = client.list\_collections()  
 print(f"✅ ChromaDB: {len(collections)} colecciones")  
 except Exception as e:  
 print(f"❌ ChromaDB: Error - {e}")  
   
 print("\n🎯 ESTADO FINAL:")  
 print("El ambiente está listo para ejecutar el sistema RAG")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 verify\_environment()

Ejecutar verificación:

python verify\_environment.py

## C.12 Soporte y Contacto

Para problemas de configuración no cubiertos en este anexo:

1. **Revisar logs** del sistema (support\_model.log)
2. **Verificar versiones** de Python y dependencias
3. **Consultar documentación** de paquetes específicos
4. **Contactar** al equipo de desarrollo con detalles del error

**Nota**: Esta configuración ha sido validada en los entornos utilizados durante la investigación experimental (julio 2025). Versiones más recientes de las dependencias pueden requerir ajustes menores.