

ניסוי מעבדה: התקנת Ollama RAG

מבנה כללי -- 4 תתי ניסוי

1 הקדמה

מטלה זו עוסקת בנייתו ולימוד של התקנות מבוססות RAG ו-Ollama. שורת הניסויים המוצעת להלן מהווה מסגרת רעיונית כללית, ואתם מוזמנים לפרש, לפתח ולחקור את הנושאים בכל דרך שתמצאו לנכון. עבור כל ניסוי עליכם להגדיר שאלות מחקר, לבצע את הניסויים ולנתח את הממצאים, רצוי תוך הצגת ניתוח סטטיסטי ויזואלי (באמצעות גרפים או טבלאות). מומלץ לחזור על כל ניסוי מספר פעמים כדי להבטיח תוקף סטטיסטי לתוצאה.

שימו לב: המסקנות שלכם אינן חייבות לחפוף לחומר שהוצג בשיעור. אתם רשאים להגיע לתובנות עצמאיות, ובלבד שתנמקו אותן היטב; במקרים אלו מומלץ להיעזר בסימוכין חיצוניים ולהציע הסבר לפערים שגיליתם. קחו את הניסויים למקום שמעניין אתכם ולכיוון החקירה האישי שלכם -- ההנחיות הללו נועדו לשמש כ'סיעור מוחות' ואינן בגדר הגדרות סגורות.

2 כלי עזר לשימוש בניסוי

ניסוי זה מיועד לחובבי ההתקנות הטכניות.
הניסוי הזה מבוסס על הסרטון מיוטיוב:

Ollama Course -- Build AI Apps Locally
<https://www.youtube.com/watch?v=GWB9ApTPTv4&t=123s>

בנוסף, מצורפת חוברת הסבר שנכתבה על בסיס הסרטון.

3 תת-ניסוי 1: "Baseline" נכון לפי הסרטון

מטרה: לשחזר 1:1 את מה שהמדריך בסרטון עושה -- כדי שיהיה "קו ייחוס".

1.3 הסטודנט מתבקש

1.1.3 להתקין ולהריץ

- Ollama (שרת מקומי כברירת המחדל על localhost:11434)
- מודל שפה בסיסי: llama3.2 או llama 3.2 3B (כפי שהומלץ)
- מודל embedding: nomic-embed-text (pull דרך Ollama)

2.1.3 לבנות RAG "קלאסי" כפי שמתואר בספר/סרטון

Chunking: שימוש ב-RecursiveCharacterTextSplitter עם chunk_size≈1200, overlap≈300 וה-separators שהוצגו.

Embeddings: שימוש ב-OllamaEmbeddings עם `.model='nomic-embed-text'`

Vector Store: `Chroma.from_documents(..., embedding_function=OllamaEmbeddings, persist_directory=...)`

Retrieval + Generation: `similarity_search(query, k=3)` ואז `prompt` בסגנון: "Answer the question based ONLY on the following context: ..."

3.1.3 להפעיל סקריפט פייתון קצר שמבצע

- אינדוקס (חד-פעמי) של מסמך קטן (למשל דף מדיניות פנימית קצר שמסופק לסטודנט)
- שאילת 2--3 שאלות פשוטות
- מדידת:
 - זמן אינדוקס (שניות)
 - זמן תשובה ממוצע
 - האם התשובה מדויקת (כן/לא, לפי פתרון ידוע)

4.1.3 הסטודנט חייב לוודא שהכול לפי ההוראות בסרטון

- אותן פקודות `ollama pull / create / run`
- אותו מודל `embeddings`
- אותה ספריית `Vector Store (Chroma)` ואותה אינטגרציה `LangChain`

2.3 תוצר: "Baseline Report" קצר

- תצורת מערכת (מודלים, ספריות, פורט, ספריית `persist`)
- זמני ריצה
- איכות תשובות

4 תת-ניסוי 2: שינוי החלטה אחת -- ומה נשבר?

מטרה: להבין רגישות ההתקנה והסטאק: שינוי קטן → השפעה טכנית/התנהגותית. הסטודנט בוחר **לפחות 2** מהשינויים הבאים (על בסיס מה שהסרטון המליץ לא לעשות / לא עשה בפועל):

1.4 שינוי מודל ה-LLM

במקום `llama3.2` → מודל קטן יותר (3B) או גדול יותר (8B).
בדיקה:

- זמן תשובה (Latency)
- שימוש בזיכרון (אם אפשר למדוד)
- איכות תשובה (אותן שאלות מניסוי 1)

2.4 שינוי מודל ההטמעה

במקום nomic-embed-text → מודל Embedding אחר (אם הסרטון/ספר מזכירים חלופה; או אפילו Embedding ענני).

בדיקה:

- איכות האחזור (האם עדיין חוזרים אותם Chunks?)
- שגיאות אפשריות בחיבור (כתובת שרת, API key, עיכוב רשת)

3.4 שינוי פרמטרים של Chunking

chunk_size קטן מאוד (למשל 003) או גדול מאוד (0003), overlap שונה.

בדיקה:

- מספר ה-Chunks שנוצרו
- איכות התשובות: האם נוצר מצב "לא נמצא בהקשר" כי המידע נחתך/הוצף?

4.4 "שוכחים" להתמיד את ה-Vector Store

לא להשתמש ב-persist_directory; כל ריצה בונה אינדקס מחדש.

בדיקה:

- זמן "אינדוקס" בכל ריצה
- תחושת שימושיות/Scalability (אינטואיציה על פרודקשן)

5.4 טבלת השוואה

הסטודנט מריץ מחדש את אותן שאלות מניסוי 1, ומתבקש למלא טבלה:

שינוי B	שינוי A	Baseline	מדד
			זמן תשובה ממוצע
			מספר Chunks
			האם התשובה מדויקת
			בעיות טכניות

6.4 מטרה פדגוגית

לגרום לו להבין את המשמעויות הטכניות של ההמלצות בסרטון: למה דווקא chunk_size כזה, למה nomic-embed-text, למה Chroma, ולמה להפריד Indexing מ-Retrieval.

5 תת-ניסוי 3: Ollama מקומי מול חלופה חינוכית בענן

מטרה: להמחיש את ההבדלים בין:

- "חירות חינוכית מקומית" (Ollama על הלפטופ שלך)
 - שירות ענן חינוכי/זול (למשל API ציבורי עם מגבלות)
- הניסוי יכול להיות גם תיאורטי/סימולטיבי אם לא רוצים באמת לחבר API, אבל עדיף דוגמת קוד קונקרטית.

1.5 הסטודנט מתבקש

1.1.5 לשכפל את פיפליין ה-RAG

להחליף את ה-LLM המקומי (llama3.2 דרך Ollama) ב-LLM ענני (למשל מודל חינמי ב-HuggingFace Inference, Groq, Together, וכו').
להשאיר את כל שאר השלבים מקומיים: Vector Store, Embeddings, Chunking.

2.1.5 להריץ את אותה שאלתה בדיוק בשלושה "מצבים"

- **מצב A:** הכל מקומי (Ollama LLama + nomic-embed-text + Chroma)
- **מצב B:** Embeddings מקומיים, LLM בענן
- **מצב C:** LLM + Embeddings בענן (אם אפשר/רוצים)

3.1.5 למדוד

- Latency (כולל רשת)
- תלות ברשת (האם אפשר לרוץ offline?)
- תקלות אותנטיקציה/Rate Limit
- איכות התשובה (צריכה להיות דומה)

2.5 המסקנות הרצויות

- להבין את tradeoff: עלות/זמן/פרטיות/פשטות התקנה
- לראות שהארכיטקטורה עצמה (RAG) זהה, רק ה-"מנוע" מתחלף

6 תת-ניסוי 4: "סוגי RAG" וגישות מתקדמות

מטרה: לקשור את התאוריה מהפרקים -- סוגי כשלי RAG ואסטרטגיות Context -- למימוש טכני. הסטודנט יתנסה לפחות בשתי וריאציות:

1.6 Basic RAG vs Contextual Retrieval

1.1.6 Basic

Embedding כל Chunk כמו שהוא.

2.1.6 Contextual Retrieval (בהשראת Anthropic)

לפני ההטמעה, יצירת "כותרת/תיאור הקשר" לכל מסמך (ע"י LLM), צירוף התיאור ל-Chunk ואז Embedding.

3.1.6 השוואה

- האם שאלות "גבוליות" (למשל ניסוח מרומז) מצליחות יותר בשיטה השנייה?
- מדידה: כמה מתוך N שאלות חוזרות עם Chunk "נכון"

2.6 Basic Retrieval vs Reranking

1.2.6 Retrieval רגיל

→ similarity_search(k=5) זריקה ישירה ל-LLM.

2.2.6 Reranking

→ similarity_search(k=10) שימוש ב-LLM (מקומי או ענני) לרה-דירוג לפי התאמה לשאלה → הכנסת רק טופ 3 ל-LLM לשאלה הסופית.

3.2.6 בדיקה

האם יש פחות כשלי "Missed Top Rank / Not in Context" כפי שתוארו בפרק ה-anti-patterns.

3.6 מטרת הניסוי

להפוך את תאור ה-anti-patterns והאסטרטגיות (chunking נכון, rerank, contextual) ממלל תיאורטי לקוד מדיד.

7 דרישות מטה-פדגוגיות

כדי להבטיח שהסטודנט באמת יצפה בסרטון:

1.7 שאלות ספציפיות לסרטון

בכל תת-ניסוי, יש "שאלה ספציפית לסרטון", לדוגמה:

- "איזה מודל Embedding מומלץ בסרטון ומדוע?" → עליו לזהות זאת מתוך הסרטון/תמלול
- "איזה פורט השרת של Ollama ומתי צריך לשנות אותו?"
- "איזה flags מופיעים בפקודת ollama create בדמו?"

2.7 צ'ק-ליסט סופי

הסטודנט צריך לצרף:

- צילום מסך/פלט טרמינל של פקודות ollama שנלקחו מהסרטון
- הסבר מילולי קצר: "מה ההחלטה הטכנית שהוידאו הציע, ומה עשיתי אחרת בתת-ניסוי 2/3/4?"

8 סיכום הניסוי

הניסוי קצר (מסמך קטן, מעט שאילתות) אך עשיר בוריאציות מימוש.

1.8 כל תת-ניסוי

- משתמש באותו pipeline בסיסי (Ollama+LangChain+Chroma+Embeddings)
- משנה רכיב/פרמטר אחד ומודד השפעה טכנית (Latency, שגיאות, RAG failures)

2.8 התוצאה

הסטודנט לא רק "יודע מה זה RAG ו-Ollama", אלא מבין:

- איך התקנה/קונפיגורציה משפיעות
- מה האלטרנטיבות המקומיות/ענניות
- למה ההמלצות בסרטון/ספר הן כאלה, ומה המחיר של לסטות מהן

3.8 הצגת התוצאות

על הסטודנט לתכנן ולחשוב באיזה אופן משכנע להציג את תוצאות הניסוי, החקר של הניסוי, והמסקנות מהניסוי. מומלץ לתקף את התוצאות בגרפים לפי שיקול דעת הסטודנט.

הערה: מסמך זה כתוב בלשון זכר מטעמי נוחות בלבד, אך מיועד לנשים ולגברים כאחד.