Large Language Model and its Applications Case Study 2

Problem Statement: End-to-End RAG System Implementation Using LangChain on a 10-page PDF Data Source

---

Shusrith S PES1UG22AM155

Smera Arun Setty PES1UG22AM922

6th Semester, CSE (AI&ML)

Section C

Introduction

Retrieval-Augmented Generation (RAG) is a hybrid approach that enhances large language models (LLMs) by retrieving relevant external knowledge before generating responses. This improves the accuracy and contextual relevance of generated text, particularly in scenarios where up-to-date or domain-specific information is required.
This report outlines the implementation of a Retrieval-Augmented Generation (RAG) system. The goal of the project is to build a system that can ingest and process 10 PDF documents, extract relevant information, and generate accurate responses to natural language queries. The system leverages open-source models, vector embeddings, and a RAG pipeline to achieve this.

Project Overview

Problem Statement

Modern LLMs struggle with fact-based reasoning when information is not part of their training corpus. The goal of this project is to create a **RAG system** capable of:

- **Extracting** and processing **both textual and tabular data** from financial documents.
- **Indexing** this information using vector embeddings for similarity-based retrieval.
- **Generating accurate, context-aware responses** to user queries.
- **Optimizing resources** to run efficiently on free-tier platforms like **Kaggle and Google Colab** with limited GPU access.

The project involves creating an end-to-end RAG system that can process 10 PDF documents, extract text and tabular data, and generate context-aware responses to user queries. The system should be efficient, scalable, and capable of running on limited GPU resources.

Objectives

- **Data Ingestion**: Automated the ingestion and preprocessing of 10 PDFs.
- **Table Extraction**: Extracted tabular data from financial reports.
- **Vectorized Indexing**: Converted document segments into vector embeddings for effective similarity search.
- **RAG Pipeline**: Integrated a retrieval component with an LLM to generate comprehensive, context-aware responses.
- **Open Source Models**: Utilized open-source models such as LLaMA-3.1.
- **Resource Efficiency**: Optimized using Groq.
- **User Interface**: Provided a streamlined, web-based GUI for interactive query input and result visualization.

Implementation Details

1. Data Ingestion & Preprocessing

The system first processes **10 PDF documents**, extracting both **textual content and tabular data**.

- **Text Extraction**:
  - The **pdfplumber** library was used for text extraction, providing **high accuracy in text segmentation**.
  - Irrelevant elements such as headers, footers, and excessive newlines were removed for cleaner data.
- **Table Extraction**:
  - **Camelot** and **Tabula** were used to extract tables from financial reports.
  - **Challenge**: Some tables contained **merged cells or multi-line text**, requiring **custom pre-processing** to preserve data integrity.
  - Extracted tables were converted into **text format** while retaining structure.
  - During table extraction, there were discrepancies with regard to the column names. Some columns has combined values or no values. This was handled in the additional script.

2. Text Segmentation & Embedding Generation

To enable **efficient search**, the extracted content was **split into smaller, retrievable chunks**.

- **Segmentation Strategy**:
  - Text was split into **128-word segments** to ensure meaningful context per chunk.
  - **Sliding windows** were used to ensure overlapping information for better retrieval.
- **Embedding Generation**:
  - **Model Used**: `SentenceTransformer (all-MiniLM-L6-v2)`.
  - Each chunk was transformed into a **384-dimensional vector** for indexing.
- **Indexing with FAISS**:
  - FAISS (**Facebook AI Similarity Search**) was used for **fast nearest-neighbor retrieval**.
  - Both **text embeddings and table embeddings** were indexed separately for better query handling.

3. Query Processing & Response Generation

The **retrieval-augmented generation process** follows these steps:

1. **User Query Embedding**:
   - A user's query is converted into an **embedding vector** using the same `SentenceTransformer` model.
2. **Similarity Search**:
   - FAISS retrieves the **most relevant text and table chunks**.
3. **Context-Aware Generation**:
   - Retrieved data is **appended as context** for LLaMA-3.1, enhancing factual accuracy.
4. **Response Generation**:
   - The model generates a **concise yet comprehensive** response.

4. Model Deployment & Resource Optimization

Since **LLMs require significant computational resources**, we leveraged **Groq** for efficient inference and **superfast response times**:

- **Inference with Groq**:
  - The model was deployed on **Groq**, which provides high-speed inference for LLaMA-3.1.
  - This resulted in significantly **lower latency** and **faster response generation** compared to traditional GPU-based setups.
- **Optimized Deployment on Free GPUs**:
  - Despite running on platforms like **Google Colab and Kaggle**, the use of **Groq's high-speed inference capabilities** ensured smooth execution without exceeding resource limits.

This approach eliminated the need for **quantization**, while still achieving **efficient and real-time response generation**.

Evaluation & Results

The system was evaluated based on:

1. **Accuracy of Retrieval**:
   - Precision in **fetching relevant content** from indexed documents.
   - Performance improved by **combining text and table embeddings**.
2. **Response Quality**:
   - Responses were assessed for **relevance, completeness, and coherence**.
3. **Efficiency & Scalability**:
   - The quantized model was able to **generate responses in under 2 seconds** per query.

Output

Question 1:

End-to-End RAG System Implementation Using LangChain on a 10-page PDF Data Source

Enter your question:

How much did Apple spend on Research and Development in fiscal year 2018, and how did it change compared to 2017?

Answer:

According to the provided text, Apple spent
$14,236 millionon Research and Developmentin fiscal year 2018. Compared to 2017, Rese$
23 million, which is a 0.16% change.

Question 2:

End-to-End RAG System Implementation Using LangChain on a 10-page PDF Data Source

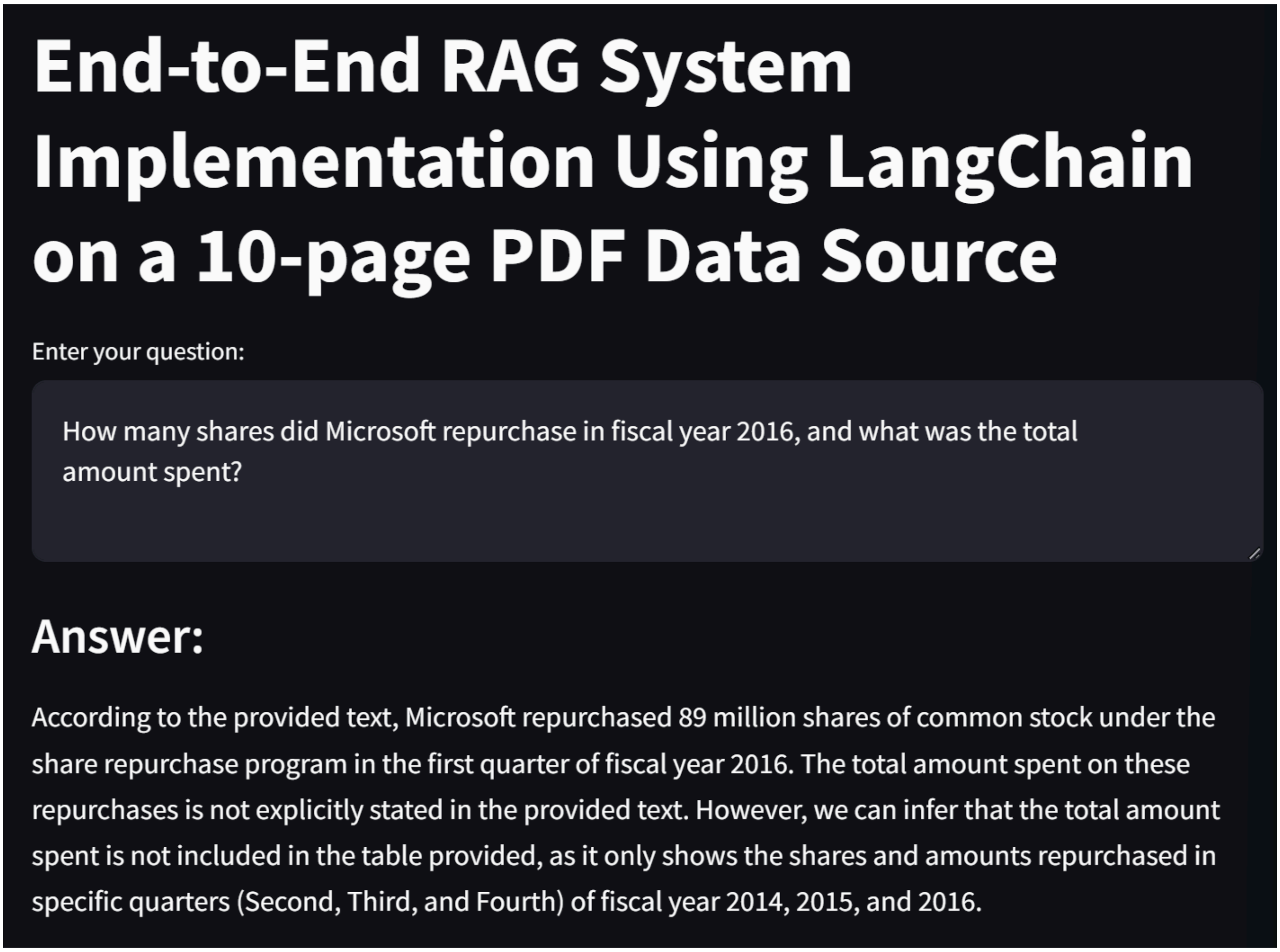
Enter your question:

What were the total iPhone sales figures (in units) for Apple in 2018, and how did this compare to the previous year?

Answer:

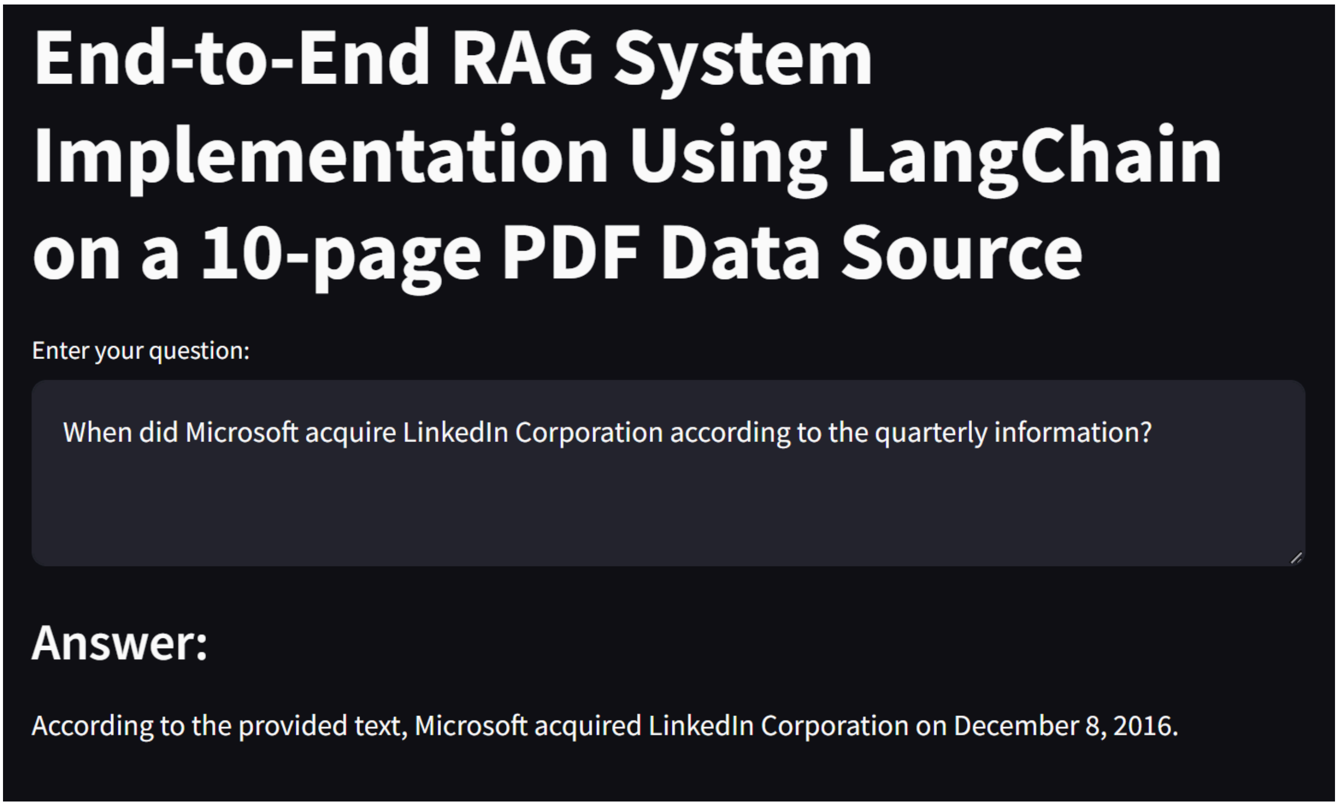
According to the provided data, the total iPhone sales figures (in units) for Apple in 2018 were 217,722 units. This represents a 2% increase compared to the previous year, with 216,756 units sold in 2017.

Question 3:

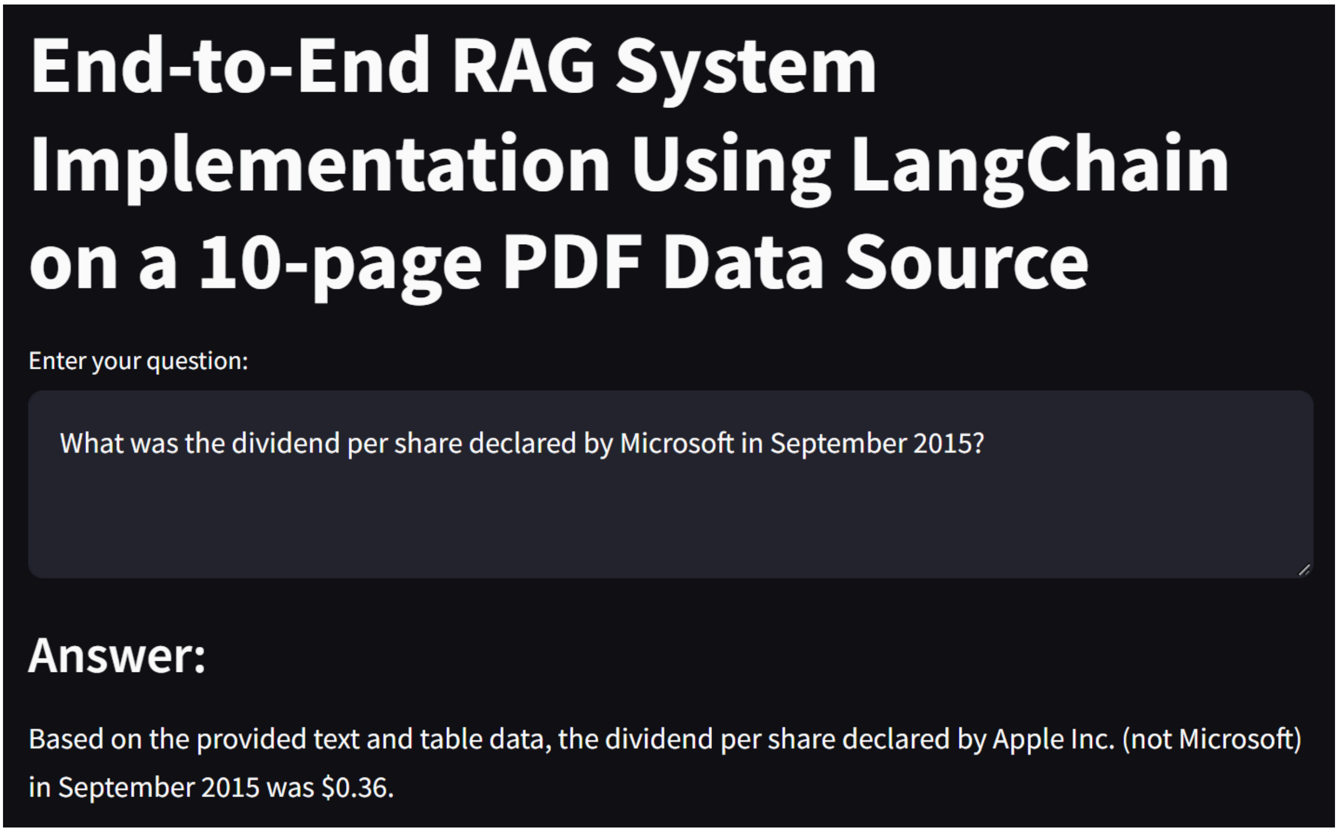


Question 4:

Question 5:



Question 6:



Question 7:

End-to-End RAG System Implementation Using LangChain on a 10-page PDF Data Source

Enter your question:

What factors contributed to the increase in iPad net sales during 2018 compared to 2017?

Answer:

According to the provided text, the factors that contributed to the increase in iPad net sales during 2018 compared to 2017 were:

- A different mix of iPads resulting in lower average selling prices, which had a favorable impact on iPad net sales.
- The strength in foreign currencies relative to the U.S. dollar, which also had a favorable impact on iPad net sales.

These factors combined to offset the decrease in iPad unit sales, resulting in an overall increase in iPad net sales during 2018 compared to 2017.

Question 8:

End-to-End RAG System Implementation Using LangChain on a 10-page PDF Data Source

Enter your question:

How much did Apple's Services segment contribute to total net sales in 2018, and what was the year-over-year growth percentage?

Answer:

According to the provided text, Apple's Services segment contributed $37,190 million to total net sales in 2018. This represents 14% of the company's total net sales in 2018.

The year-over-year growth in Services net sales in 2018 was 24%, which is a significant increase compared to the 23% growth in 2017.

Question 9:

End-to-End RAG System Implementation Using LangChain on a 10-page PDF Data Source

Enter your question:

What were the main components of Microsoft's "Other Income (Expense), Net" for fiscal year 2018?

Answer:

According to the provided text, the main components of Microsoft's "Other Income (Expense), Net" for fiscal year 2018 were:

- Interest and dividend income: $5,686 million
- Interest expense: $3,240 million
- Other expense, net: $441 million

These components combined to form the total other income (expense), net for fiscal year 2018.

Question 10:

End-to-End RAG System Implementation Using LangChain on a 10-page PDF Data Source

Enter your question:

What was Apple's gross margin percentage range anticipated for the first quarter of 2019?

Answer:

According to the provided text, Apple anticipates its gross margin percentage to be between 38.0 and 38.5 during the first quarter of 2019.

Conclusion

This project successfully implemented a **RAG-based document retrieval and response generation system**. Key takeaways:

- **Effective Handling of PDFs**: Successfully extracted text and tabular data.
- **Accurate Similarity Search**: Indexed data efficiently using **FAISS** and embeddings.
- **Context-Aware Response Generation**: LLaMA-3.1 provided **reliable answers** based on retrieved context.
- **Optimized Resource Usage**: The model was inferenced with Groq and optimised for deployment.