Michelin Digital and IT operations are based in Pune, Maharashtra. The team at this location works on cutting edge technologies such as AI, IOT, Data Analytics, CRM, Mobile Apps, APIs, UX/UI Design, E-Retail, Mapping, Cloud, Office 365, Oracle DBS, ServiceNow and others ready for the future. technologies.

TireScan

Manager: Noel Athaide

April 30, 2024

Abstract

This report details the development of TireScan, a system designed to extract information from web data using LLMs based on European Article Numbers (EANs) of tires. TireScan first scrapes relevant web pages associated with the provided EAN. Then, it preprocesses the scraped data and uses LLMs to validate the EAN and extract useful information like tire brand, diameter, model, and vehicle type. This report explores four different algorithms for information extraction and compares their performance.

Introduction

Michelin, a global leader in tire production, strives to leverage innovative technologies to improve efficiency and customer experience. TireScan aims to address the need for a quick and reliable method to extract tire information from online sources. By analyzing web data using LLMs, TireScan automates the information retrieval process, saving time and reducing manual effort.

Project Objectives

The primary objectives of this project were:

- Develop a system to extract tire information from web data using LLMs.

- Explore and compare different algorithms for information extraction.
- Analyze the performance and limitations of the algorithms.
- Create a comprehensive report documenting the project development and findings.

Project Methodology

The project involved the following stages:

- **Data Collection:** Utilized the Google Custom Search API to scrape relevant URLs associated with the provided EANs.
- **Data Preprocessing:** Cleaned and preprocessed the scraped data by removing unwanted tags, extracting relevant tags, and translating the text if necessary.
- **Model Selection:** Explored LLMs (OpenAI and Gemini APIs) to validate the EAN and extract information.
- **Algorithm Development:** Implemented four different algorithms:
    - RAG Pipeline: Retrieval-Augmented Generation using vector store and LLMs.
    - Prompt Updating: Iterative refinement of prompts based on context.
    - Langchain's Refine: Utilizes Langchain's Refine chain for continuous refinement.
    - Langchain's Extraction Chains: Leverages Langchain's extraction chains based on a defined schema.
- **Evaluation and Analysis:** Evaluated the performance of the algorithms based on accuracy, efficiency, and ease of implementation.
- **Report Generation:** Documented the project methodology, results, and conclusions in this report.

System Architecture

TireScan is composed of the following key components:

- **Scraper:** Uses Google Custom Search API to find relevant URLs for the provided EAN.
- **Preprocessor:** Cleans and preprocesses the scraped data by removing unwanted tags and extracting relevant tags.
- **Validator:** Employs LLMs to determine the validity of the EAN.
- **Extractor:** Extracts tire information using the selected algorithms.

Algorithm Description

RAG Pipeline

This algorithm leverages a retrieval-augmented generation (RAG) pipeline to extract information:

1. **Vector Store Creation:** Embeddings are generated for preprocessed text chunks using the Azure OpenAI Embeddings model, and these embeddings are stored in a vector store (Chroma).
2. **Retrieval Chain:** A retrieval chain is defined to search the vector store based on queries related to specific tire attributes. This returns relevant context.
3. **LLM Processing:** An LLM (Gemini-Pro) processes the retrieved context to generate the answer for the attribute.

Prompt Updating

This algorithm utilizes an iterative prompt refinement technique:

1. **Chunking:** The preprocessed scraped data is split into smaller chunks.
2. **Prompt Iteration:** An LLM (Gemini-Pro) is queried with each chunk, using the previous chunk's output as additional context for the next query.
3. **Output Refinement:** The LLM updates the previous output based on the new context, refining the results.

Langchain’s Refine

This algorithm employs Langchain's *Refine* chain for continuous refinement:

1. **Prompt Templates:** Prompt templates are defined to extract information and refine the output based on new context.
2. **Refine Chain:** Langchain's *Refine* chain is utilized to iteratively process the text chunks, updating the output based on new context.

Langchain’s Extraction Chains

This algorithm uses Langchain's extraction chains to extract information based on a defined schema:

1. **Schema Definition:** A schema is defined specifying the attributes to be extracted and their data types.
2. **Extraction Chain:** Langchain's extraction chain is employed to extract information from the text based on the defined schema.

Conclusion

The RAG pipeline achieved high accuracy but required significant time for processing due to vector store creation and search. The prompt updating method showed good accuracy and efficiency but required careful prompt engineering. Langchain's Refine was simpler to implement than the RAG pipeline but had lower accuracy. Langchain's extraction chains provided a straightforward way to extract information but could struggle with complex text structures.

TireScan successfully demonstrates the potential of LLMs for extracting tire information from web data. The RAG pipeline exhibited the highest accuracy, but prompt updating offered a balance between accuracy and efficiency. Langchain's Refine and extraction chains provide convenient approaches for simpler extraction tasks.

Future Work

Future work will focus on:

- Improving the accuracy of the RAG pipeline.
- Developing more robust methods for handling complex text structures.
- Integrating TireScan with a database to store extracted information.
- Building a user interface for easy interaction with TireScan.