Ryan Laird

3/12/2025

Creative Problem-Solving Exercise

Problem

One of the world's largest custom furniture manufacturers is struggling with efficiency. They serve large design/architecture firms and have a direct-to-consumer business. The problem:

- 300+ suppliers and tens of thousands of SKUs (Wood, Plywood/MDF, Veneer, Legs, Hardware, Fasteners, Cabinetry, Stain, Paint, Lacquer and Varnish, Foam, Fabric, Webbing and Springs, Batting, Glass and Mirrors, Metal Frames and Panels, etc)
- Customers will come to them with concept sketches and/or detailed descriptions of requirements
- Many suppliers provide configurator tools to build final products from their individual catalogs, but there is no efficient way for the manufacturer to search and configure across multiple suppliers
- Heavy reliance on their design team's experience and knowledge to recall and create a bill of materials from memory

They're asking how generative AI can help. Please provide:

1. Your approach to analyzing and solving this problem
2. A technical proposal, including specific technologies and implementation strategies

My proposal:

Step 1, Layout the Problems Presented:

- Their bill of materials is being created manually (Dependent on human memory).
- Their supplier chain is complex and there are large amounts of SKUs.
- Each supplier has their own configurator tools, but there is no efficient way for the manufacturer to search and configure across multiple suppliers.

Step 2, Determine and Layout the Solutions to the Problems Presented:

1. Create a **unified relational database**. Develop multiple APIs to pull data from the supplier configurator tools. This allows all supplier data to reside into one central database.
2. Using **Natural Language Processing models**, develop a multimodal AI interface. This interface will convert concept sketches and or detailed descriptions of requirements to structured data. Customer uploaded sketches would receive AI generated suggestions for compatible materials and designs.
3. Develop an Automated AI powered Bill of Materials bot using a **large language model**-based system. Upon each finalized design request submission, the bot will generate an optimized Bill of Materials. This bot would: optimize cost, availability, and sustainability. Furthermore, it would provide alternative selections in case first option materials are unavailable.

Step 3, Implementation Steps:

1. Use REST and GraphQL APIs to pull in configurator data.
2. Convert SKUs into a unified, structured format.
3. Train CLIP (Vision Model) to parse sketches. Use Claude (Natural Language Processing) for text descriptions.
4. Using Neo4j, Implement Graph Based Supplier search.
5. Use Large Language Models and Graph Neural Networks for optimized material selection.
6. Use Retrieval Augmented Generation and API updates to pull real-time supplier data.
7. Use a Hybrid AI interface (cloud and edge computing) for load balancing (Azure) with FastAPI backend.

Step 4, Layout Technology Stack:

Database: *PostgreSQL (Relational Database)* for managing supplier catalogs and standardized SKU data. *NEO4j* for graphing materials catalog relationships.

Large Language Model: Train CLIP with Contrastive Learning to match sketches with similar existing designs and suggest relative materials. Also, train CLIP with furniture specific datasets to increase model accuracy. Augment LLM with Graph Neural Networks to analyze supplier relationships and material compatibility. Implement Retrieval Augmented Generation. This causes AI to pull real time supplier data before generating a Bill of Materials.

Building Interactive Web User Interface (Frontend): React.js full stack application where customers or designers can search and configure materials.

Building Interactive Mobile User Interface: React Native to expand user platform to mobile devices.

API Framework: REST APIs, GraphSQL, or WebSockets to pull data from supplier configurator tools and facilitate seamless communication between the frontend and backend. Use REST APIs for standard supplier data retrieval. Use GraphQL for dynamic queries across multiple suppliers. Use WebSockets for real-time supplier updates.

Backend: FastAPI to handle backend of REST APIs.

Hybrid AI interface: AZURE for cloud infrastructure. For mobile, use TensorFlow Lite for cloud storage.