A chatbot for PartSelect.com

Some notes + implementation details – Sharan Patil

To Ensure The Project Runs

```
pip install -r requirements.txt
```

```
npm install # should install all the react libraries in packages.json
```

```
npm install react react-dom # incase packages.json is missing
```

```
npm install lucide-react
```

```
npm install @vitejs/plugin-react
```

```
npm install vite
```

Then run the project as shown in the video presentation.

Initial attempt at making the chatbot – a WebGPT inspired approach

Building an Agent to Browse PartSelect

I had this idea after doing some research and reading up about WebGPT, a language model that can search the internet, navigate web pages, follow links, and cite sources.

After browsing PartSelect, I realised that a lot of the common user queries had their answers on the website itself.

I thought it would be best to try and build an agent (calling on the DeepSeek API) that could browse PartSelect and quickly find what the user is looking for. This is what my initial attempt of the chatbot was.

Problems Faced and Why it Failed

One issue I faced was that PartSelect had protocols in place that forbid bots from browsing the site. In order to navigate around this, DeepSeek had to introduce human-like delays while it was browsing, increasing run-time significantly. It would often fail at accessing the site multiple times before finally breaking through and being able to access it.

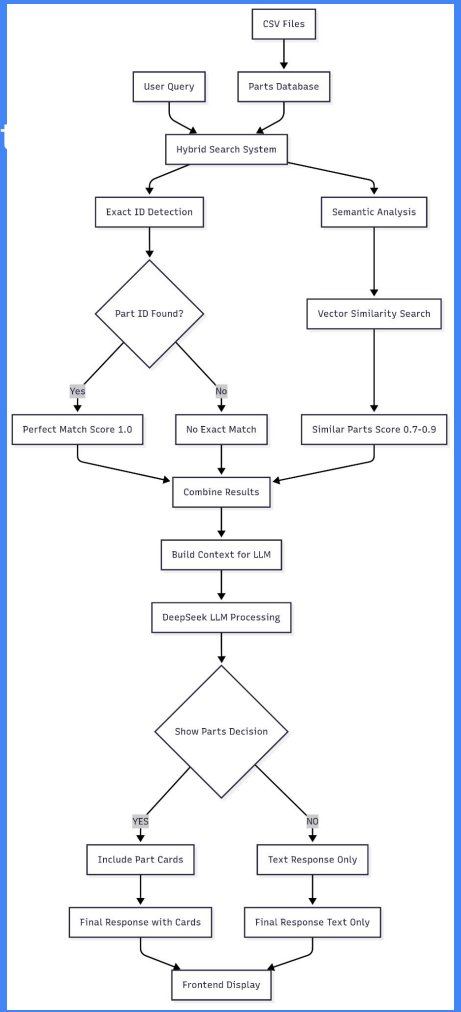
But the biggest reason by far was the fact that DeepSeek could not detect the presence of the search bar on the PartSelect website – and even after hard-coding it to recognise PartSelect's search bar specifically, it could not use the search functionality correctly, no matter how I tried prompting it. Since DeepSeek couldn't use the search bar, it was effectively useless, and was wasting minutes simply trying to access the site. I realised it would likely need specialised model training to get DeepSeek to browse effectively, and thus retired this approach after a day.

Final Choice – RAG-based agent

The diagram shows a flow chart of the RAG-based agent. A Hybrid Search System was put in place to fix errors when prompting the chatbot with any query involving part IDs – oftentimes, DeepSeek would say that a certain part ID could not be found in the database, even though it was actually present. This is likely due to the part IDs having a poor representation in the word embedding space. In the hybrid search system, if an exact part ID match is found in the database, the data associated with this part ID is directly taken from the database and given to the LLM as context, and the answer to the query is generated by the LLM agent from there.

In order to get a database for RAG, a web scraping program was implemented to scrape information about the refrigerator and dishwasher parts from PartSelect.com. The program was written to scrape data from all links in the “Related {Appliance} Parts” section of this webpage (and similarly for refrigerators).
The program was able to scrape information of a decent number of parts, but far from all of the parts on the website (I decided to scrape this way due to time constraints).

The “Part Card” decision tree at the end of the flowchart is just the model deciding if it should recommend similar products or not, based on context.



To Generate/Scrape the Datasets

In order to generate the `refrigerator_parts.csv` and `dishwasher_parts.csv` files, simply run the following command:

```
python3 scrape_data.py
```

Thank You!