Auto Guide Application

Team-10

Pramod Kalidindi
Graduate Student,
Arizona State University.
pkalidi1@asu.edu

Prashanth Radhakrishnan
Graduate Student,
Arizona State University.
pradhak3@asu.edu

Satish Bhambri
Graduate Student,
Arizona State University.
sbhambri@asu.edu

Vimal Menon
Graduate Student,
Arizona State University.
vmenon7@asu.edu

***Abstract*— This paper describes about the imminent need for a good semantic web application with proper statistical data to get the details and comparison of different automobiles. It also tells about the various procedures and components of semantic web that we used to build this application.**

Keywords— SPARQL, Semantic web, Fuseki, data link, RDF

I. INTRODUCTION

In the ever and fast changing world of technology, there are a lot of websites about automobile pricings, details, reviews and, selling of used cars. Albeit these much websites and details about the automobiles, still people face a lot of dilemmas and problems regarding buying and selling automboiles, be it a fair price, proper specifications, salvaged title or establishing authenticity of the corresponding dealer. This attributes to the fact that, although the above mentioned websites are unique in their own way, they lack one or more pivotal criterias discussed above to select the best car based on the user's choice of preference. Also, these websites don't display the details based on all user preferences. To counter this, we need a good web application for automobiles, that compares the cars on all criterions, and gives the best result, based on the user' preference.

Automobile guide is one such application, that serves as a final stop and answers all the queries and doubts that were discussed previously. It works statistically to scrape the required data regarding the history of vehicles, their technical details and fundamental reviews and expert analysis of the automobiles. A very good aspect of the autombolie web application is that it combines data from the different domains of Automobile industry. It gets data from Government data Statistics for crime statistics and Autombolie tracking, end user reviews, and classifields platforms, which typically had a lot of information on the car details and reviews. We then form an ontology using the datasets we gathered from different domains, and refined the datasets using Google refine, and then generated the Resource Description Framework(RDF) using the refined data and the ontology for establishing meaningful data links, and hence, effective retrieval.

Auto Guide application works based on the well-researched, different datasets provided by the federal government, linked semantically, thereby establishing the authenticity of the data and the vehicle, one is looking for. Automobile guide serves the end user, be it a naïve user or expert technician, in the following aspects discussed below.

First and foremost, Auto guilde helps us to know the fair pricing of the autombolies we are looking for, and hence prevents any fradulent transactions at various classifield sites. Secondly, it provides us with a facility to check for the details of the car we are planning to buy or sell. It also helps us in making our decisions better by providing the important reviews for the specific model, built and year, based on some of the technical parameters such as engine displacement to provide us with the most accurate data and estimate of our purchase and sell. It provides us with the suggestions of the automboile, based on the price range we give(minimum and maximum price value).

II. GOALS

The main goal of this project is to provide users with a website about cars where they can get details on new and used cars, price based search, car comparisons within the same page, which is very essential for the users, as they find it very easy to gather information, which is unique to this application and not present in the existing applications. The other goal is to provide users with the reliable, unbiased and accurate details about cars and its different specification.

III. RELATED WORK

A. Newcars

The Newcars.com is an official website which contains details of the new cars, their pricings, and they also have the feature of giving the zip code based car search, which is cool. They have also categorized cars based on Hatchbacks, Sedans, and SUVs, which is very easy for user to search for a car.

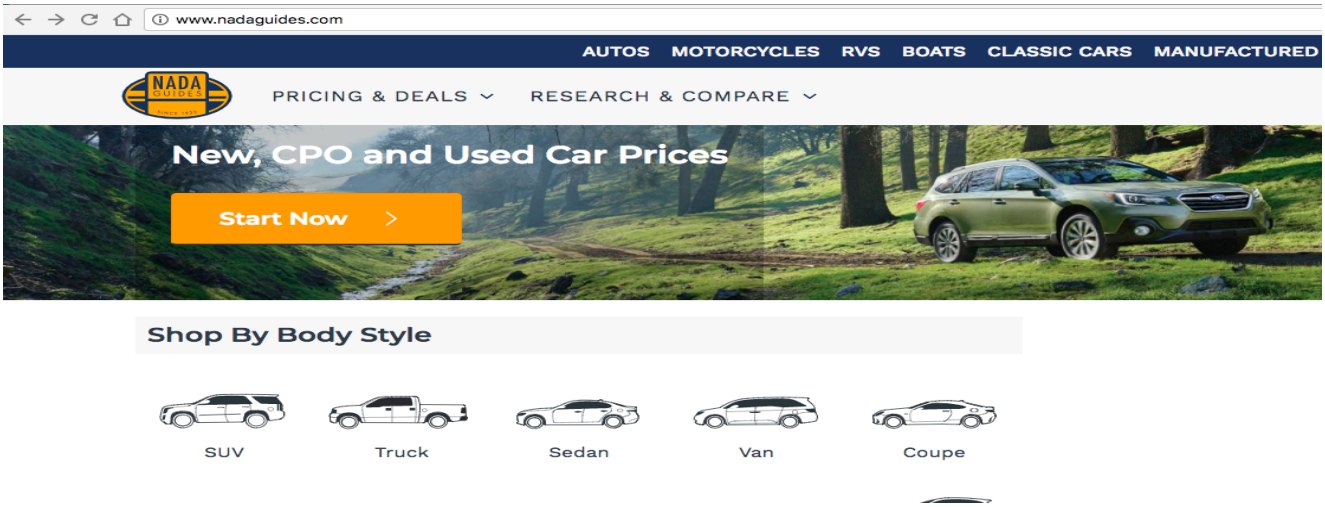

Fig 1. Image of the nadaguides.com discussed below

The Auto Guide application is better than Newcars.com in the following ways:

1. The Auto Guide provides search details for both new and used cars, whereas the Newcars provides only for the new cars.
2. The Auto Guide provides comparison among different cars, but the Newcars provides only the specification details about the car.

The limitations of Auto Guide in comparison with the Newcars.com are the following:

1. The Newcars provides the car price based on the state and zipcode, in contrast to the Auto Guide which doesn't provide car price based on the state and average income.
2. The NewCars has a field for consumer feedback and review, which is a future work for Auto Guide application.

B. Nadaguides

The National Automobile Dealers Association(NADA) is an American trade organization which provides all car details, comparison among cars and they have deals pertaining to some locations. It has very good information about car deals, car price search and other details.

The main limitations of Auto Guide in comparison with the NADA website is that:

1. They give information about the insurance centers, vehicle history report, calculate monthly payments, in comparison to the Auto guide application which don't provide that information.
2. The NADA website has a custom specific feature selection for the cars, which is missing in the Auto guide application, and is a future enhancement for the

IV. Semantic Data Model

The Auto Guide application uses semantic data model for building the application. The multiple datasets from different domains are connected by interlinking the class and properties of all the datasets.

The key things that are addressed in building the semantic data model are the following:

1. Challenges faced in integrating the data
2. Generation of RDF instance data
3. Querying the linked data, linked from different domains using SPARQL

Let's discuss how we handled all these steps below:

A. Challenges faced in integrating the data

We collected data from different domains of the automobile including its details, specifications, reviews and ratings, and comparison among different cars. Since we got data from multiple data sources like Kaggle, DBpedia, and federal Government data, linking them was a humongous task, as we needed to filter and refine all these datasets before linking them. We also had the scrapped data from CraigsList which contained a lot of authentic information regarding the automobile specifications. We did the following for integrating the datasets:

1. Removed the redundant data and unified the column names with same content.
2. Refined the dataset using Google refine and generated the RDF.

The primary reason for using Google refine for filtering is that Google refine offers many extensions and plugins for generating the RDF. It also helped in linking datas from different data sources like Kaggle, DBpedia, and federal Government data. Moreover, adding the ontology and linking the data using the given ontology classes and properties was very easy with the RDF skeleton. Once we edit the skeleton and make all our changes, we get the corresponding RDF in xml or turtle format. Since we had a collaborative team of four members in the project, we divided the datasets equally and generated the RDF instance files for the datasets.

```
PREFIX auts: <http://www.vehicularguide.com/ontologies/automobiles.owl#>

SELECT DISTINCT ?mfrname ?model ?exterior ?interior ?comfort ?liability ?build ?overall WHERE{
  ?node auts:hasMfrName ?mfrname.
  ?node auts:hasModel ?model.
  ?node auts:basedOnInterior ?interior.
  ?node auts:basedOnExterior ?exterior.
  ?node auts:basedOnBuild ?build.
  ?node auts:basedOnComfort ?comfort.
  ?node auts:basedOnLiability ?liability.
  ?node auts:basedOnOverallRating ?overall.

} LIMIT 50
```

Fig 2. A sample SPARQL query run in Fuseki server

B. Generation of instance data

For generating the RDF instance data, we used Google Refine 3.4. It had the RDF plugin, which was useful to export the data as RDF/XML or Turtle format.

We linked the existing OWL file, by adding it as a prefix while generating the RDF skeleton. This OWL file had all the classes and properties, that are needed for building the RDF skeleton.

C. Querying the linked data using SPARQL

Once, the RDF had been generated by the Google Refine, we used the Apache Fuseki server [10] to upload the RDF data and use the SPARQL endpoints provided by the server to query the RDF data. Since we query data from multiple datasets, we need to write the query appropriately with the necessary column names we require, and applying filter and other functionalities provided by SPARQL to filter the data according to our need. This data is then presented to the user in the frontend.

V. KEY FUNCTIONALITIES OF THE APPLICATION

Auto Guide is purely a web application designed using proper semantic web standards, and it answers all the queries related to fair pricing of automobiles, facilities for the car we are planning to buy or sell, comparison among different models of the car, and get a list of cars from minimum price range to maximum price range.

A. Key Features

The Auto Guide application provides the following new features in comparison to the existing applications such as:

1. By giving the manufacturer name, model and year of the car, it gives the user a list of all specifications about that car. The user is also given an option to choose whether he wants to know about the used car or the new car.

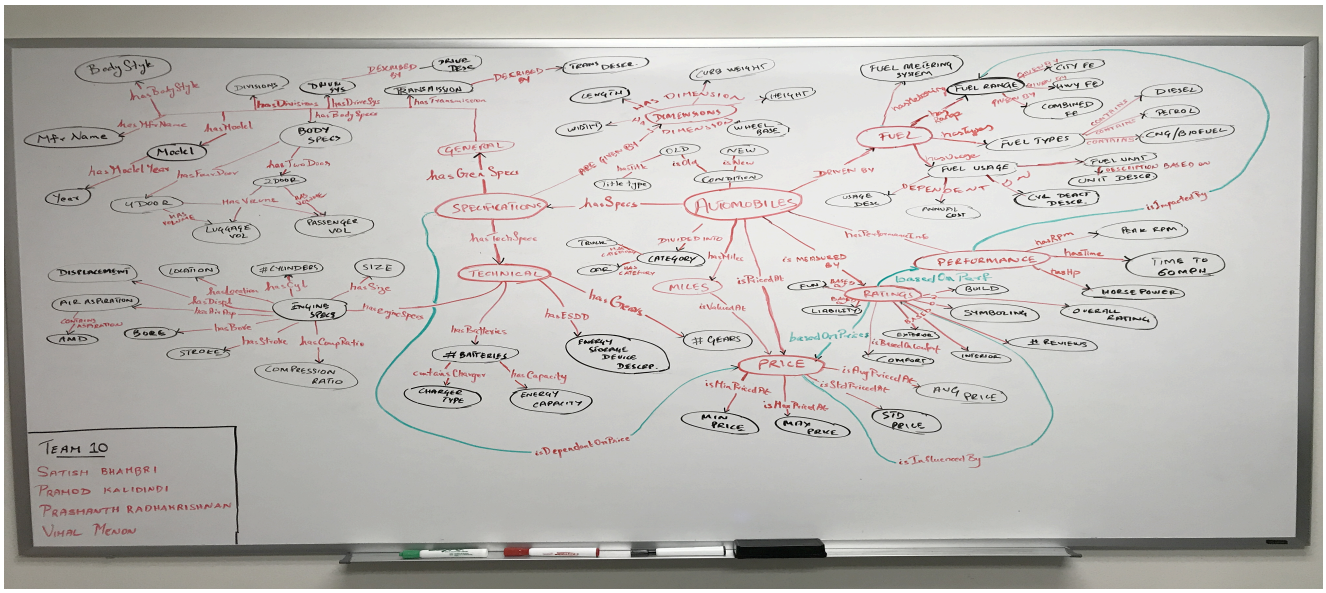

Fig 2. Ontology for Auto Guide

2. Performance: By giving the manufacturer name, model and year of the car, we can get the performance related attributes like HorsePower, Peak RPM, and so on.
3. Specifications: This feature of the application allows to get both the general specs as well as the technical specs related to an automobile.
4. Review cars: This feature allows us to get the reviews and ratings for each car and give the overall rating taking into consideration of all the review ratings.
5. Compare cars: This is one of the coolest feature of our application wherein we get the car make, model and year related to three different cars, and get the comparison among the three cars. This is very handy for users, as it will them choose the best car based on all the criterions.
6. Price based search: In price based search, we get the minimum and maximum price range for searching the cars in those price range.

B. Tools Used

1. Protégé – for Ontology generation having class and properties, and how they are inter-linked.
2. Google Refine 3.4 – Mainly used for data pre-processing, data filtering and generation of RDF instance, either in RDF/XML for mat or RDF turtle format.

C. Technologies Used

1. Apache Jena Fuseki server [10] – A server for uploading the RDF datasets and it provides the SPARQL endpoints for querying the datasets.
2. Python FLASK – It's a python framework we have used, for building the backend of the Auto guide application. It communicates to the Fuseki server [10] via the SPARQL endpoint.
3. Scrapy web Spider – Scrapy web spider is the tool used for scraping the craigslist data for the available used cars.
4. Frontend tools – Used HTML, CSS, JavaScript and Bootstrap for building the frontend part of the application.

D. Datasets Used

Datasets for Fuel Economy (US Federal Government datasets):
https://www.fueleconomy.gov/feg/download.shtml [1]
https://catalog.data.gov/dataset/fuel-economy-data

Kaggle (Used Cars dataset):
https://www.kaggle.com/orgesleka/used-cars-database

Scraped Data(Craigslist):
http://blazingseollc.com/blog/scrape-data-craigslist/

Reviews and Ratings based on performance:
http://kavita-ganesan.com/entity-ranking-data

Performance specifications:
https://archive.ics.uci.edu/ml/datasets/automobile

US Federal Government datasets:
https://www.fueleconomy.gov/feg/download.shtml

E. Challenges Addressed

From the starting, we faced a lot of challenges right from choosing the datasets, linking them, generating the RDF instances and finally integrating the SPARQL queries with the backend and the frontend. We, as a team, worked collaboratively, and divided the tasks, so that we had proper planning of the tasks that we are going to complete at each phase. Everyone in the team worked on preprocessing of the datasets. Two people worked on the front end and getting the SPARQL queries from the Fuseki server, and the rest of them worked on implementing and integrating the frontend with the backend.

One more challenge we had was to form the SPARQL queries based on our needs. The challenge here was to form queries with greater precision because of humongous data and varieties of criteria to quench user needs.

F. Integrating the Application

One of the major challenge we had was to integrate the frontend and the backend, and run the SPARQL queries from backend to the Fuseki server via SPARQL endpoint. We chose to use Python flask for our backend, as it has a built-in development server and fast debugger.

Though having a backend to query the Fuseki server was little complicated, we went on this approach, as the dependencies were less and hence, it improved the performance of the application.

VI. GOALS ACHIEVED

1. New features which is related to user's needs- Achieved by providing search for new and used cars, price based search, car comparisons within the same page, which is very essential for the users, as they find it easy to use the application.
2. Used car details from Craigslist – Achieved by scraping the data from craigslist and displaying it in a cool frontend environment, so that users find it easy to search for the cars, which is not available in the existing applications.
3. Advanced price search – Achieved by getting the minimum and the maximum price range from the user, and displaying the car based on their cost that lies within this range.

4. Reliable, unbiased and accurate details – Achieved by collecting the data from valuable sources like Federal Government, Kaggle and DBpedia.

VII. FUTURE WORK

The current work can be extended to provide the users with suggestions based on:

1. User's location data
2. Gross estimate of average income of the households of that location.
3. A feedback column from user should be provided.

VIII. CONCLUSION

The team utilized all the concepts of semantic web in the academic project, and got a good understanding of how these technologies will be useful in building a real-time web application. The team had successfully taken care of all the goals and challenges, and tried to develop a web application that covered larger audience base. The project helped the team members understanding the major challenges of the data integration and how semantic web technologies could be useful in the modern technological world.

IX. ACKNOWLEDGEMENT

The team would like to thank Dr. Srividya Bansal for giving us the opportunity to work on the semantic project, which had lots of learning, and for allowing us to add more features which were not designed during the initial proposal. I also thank her for providing some valuable ideas and suggestions during the project presentation, which further encouraged us to come up with good features for the web application. Lastly, I would also want to thank Jaydeep Chakraborty, TA, SER594, for providing suggestions for each deliverable of the project.

X. REFERENCES

[1]https://www.fueleconomy.gov/feg/download.shtml
[2] https://catalog.data.gov/dataset/fuel-economy- data
[3] http://kavita-ganesan.com/entity-ranking-data
[4] https://archive.ics.uci.edu/ml/datasets/automobile
[5]https://www.fueleconomy.gov/feg/download.shtml
[6] https://archive.ics.uci.edu/ml/datasets/automobile
[7] http://kavita-ganesan.com/entity-ranking-data
[8]https://www.fueleconomy.gov/feg/download.shtml
[9] http://blazingseollc.com/blog/scrape-data-craigslist/
[10] https://jena.apache.org/documentation/fuseki2/

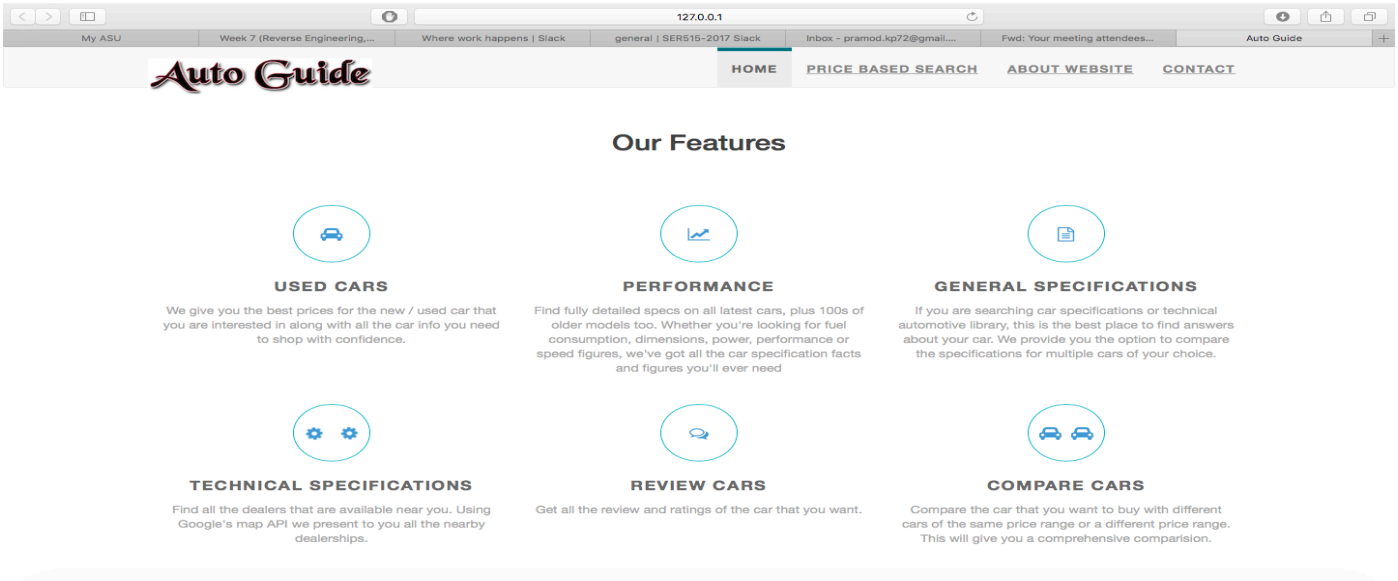

Fig 3. A basic view of our Auto Guide application