Multimedia Tools and Applications
https://doi.org/10.1007/s11042-022-13804-0

1209: RECENT ADVANCES ON SOCIAL MEDIA ANALYTICS AND MULTIMEDIA SYSTEMS: ISSUES AND CHALLENGES

Product discovery utilizing the semantic data model

Sarika Jain[1]

Received: 19 November 2020 / Revised: 27 August 2022 / Accepted: 5 September 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
Most of the existing techniques to product discovery and recommendations rely on syntactic approaches, thus ignoring valuable and specific semantic information of the underlying standards during the process. The product data comes from different heterogeneous sources and formats (text and multimedia) giving rise to the problem of interoperability. Above all, due to the continuously increasing influx of data, the manual labeling is getting costlier. Integrating the descriptions of different products into a single representation requires organizing all the products across vendors in a single taxonomy. Practically relevant and quality product categorization standards are still limited in number; and that too in academic research projects where we can majorly see only prototypes as compared to industry. This work presents a cost-effective aggregator semantic web portal for product catalogues on the Data Web as a digital marketplace. The proposed architecture creates a knowledge graph of available products through the ETL (Extract-Transform-Load)) approach and stores the resulting RDF serializations in the Jena triple store. User input textual and multimedia specifications for certain products are matched against the available product categories to recommend matching products with price comparison across the vendors. The experimental results show that semantic intelligence technologies could provide the necessary data integration and interoperability for efficient product/service discovery including multimedia.

Keywords Knowledge graph · Ontology · Engineering equipment · Multimedia data · Product categorization · Product matching · Recommendation

1 Introduction

Recommendation systems have found their usage in a multitude of applications including Information filtering, web personalization, and e-commerce. To list a few, companies like

✉ Sarika Jain
jasarika@nitkkr.ac.in

[1] Department of Computer Applications, National Institute of Technology Kurukshetra, Kurukshetra, Haryana, India

YouTube, Netflix, Spotify, Amazon, Pandora, and LinkedIn utilize the potential of recommender systems to facilitate product (videos, movies, music, products, friends, jobs) discovery. Various techniques for providing recommendations have been proposed in literature [20], most common being: the content-based, collaborative, and the hybrid approaches.

- In ***content-based*** (also called feature-based or reclusive) recommendation system, users have their preferences and items have their features; the system would correlate the target user's profile with the item's features in order to provide recommendations. Some example content-based recommendations include: "You are pursuing MCA201 course, the book XYZ is for you." and "You are a marketing guy; this new smartphone is for you".
- ***Collaborative filtering*** systems utilize the historic users' preference for items to make recommendations. They utilize some similarity algorithms to predict the best match either based on similar users' preferences or based on the ratings of similar items [38]. Some example collaborative recommendations include: "You generally buy tomato soups, here is a new flavor of tomato soup you will definitely like", "You generally buy tomato soups, here is the most popular flavor of the same", "Your friend X has bought this", and "You have just purchased paint for your house. People who bought paint also purchased a painting".
- ***Hybrid approaches*** work upon a unifying model incorporating information from both users/items' characteristics and user-item interactions, i.e., both the similarity of product attributes and the purchase history. An example of hybrid, i.e., content-based collaborative recommendation is "You just bought that book. This one is also from the same author".

The online shopping experience is evolving faster than ever before. The today's digitally empowered consumer wants seamless experience across all the touch points and expects retailers to be on the cutting edge of pricing and products. The companies are required to deliver a competitive price, a better checkout experience, immediate product availability, and above all, exact feature match of the product required. So many companies with a lot of innovative products are purging up daily, increasing the information overload with loads of multimedia data. This exponentially growing set of products confuses the customer with what to buy and from where to buy in order to quench the exact thirst and flavor of his requirement.

All the teaching and research institutes across the globe procure hardware infrastructure necessary for supporting the developmental and research activities of their projects. But due to poor understanding of the devices' technical specifications and characteristics, they face a lot of problems. With a variety of products available in the market, it becomes a daunting task understanding your options and then making the right choice. Consider simple queries like: "Why do I purchase a Network Attached Storage (NAS) when I already have file servers?", "What is the latest processor that I can afford for my new server?", OR "Should I go for Cloud Virtual Server?". There is a requirement of an interface between the customers and the retailers which can provide the best match between the triangulation of user's required specifications of the product, list of products available with those specifications, and the retailers selling those products; that could recommend to customer top n options to purchase from based on his provided specifications of the product.

Recent trends focus on the application of deep learning, exploitation of knowledge graphs, reinforcement learning, and user profiling to provide efficient explainable recommendations along with providing user privacy. The choice of strategy to be adopted depends on the strategic goal under consideration. All known approaches fail in scenarios where (a) new innovative products are introduced, (b) existing products disappear from the catalogue, (c) custom made products where each product is unique, (d) the customer has very less knowledge of the specifications, and (e) the web of data model of the semantic web.

Product Categorization Standards (PCS) like eCl@ss and UNSPSC are very common in e-commerce scenarios and are being used by many manufacturers and vendors for information exchange. These categorization standards define many product categories, their descriptions, and product property definitions and values. However, many approaches have been presented for using such standards even for the semantic web, but there have been a number of barriers for their mainstream adoption [4]. They only provide property-value pairs, meaning that the data is often incomplete and not so granular and hence not suitable for populating the target data structures. Semantic Intelligence Technologies (SITs) is the amalgamation of core AI features such as Machine Learning (ML) and Natural Language Processing (NLP), and semantic web technologies such as RDF, OWL, SPARQL etc. [21]. SITs are gaining increasing interest for representing semantic content, knowledge sharing, knowledge integration, and also automated reasoning. They have been found as good candidates for context modeling and multimedia analytics [35]. The RDF model can facilitate improving the representation of image descriptions, hence enhancing the retrieval of multimedia content with high-level semantics. Application developers can make the data meaningful and machine-understandable by defining real world entities as concepts with their properties and relationships between concepts; often termed as ontologies. These ontologies enable analysis, querying, and actions based on the semantic content and relationships, rather than simply the data points. Ontologies have proven to be very useful in the management of knowledge as they facilitate sharing of meaning among organizations and individuals. The available product categorization standards could be converted to richer representations for them to be consumed in a Linked Data and Semantic Web context. Some such ontological product categorization standards include eCl@ssOWL and GoodRelations ontology. For multimedia content (like the digital still images), we have well-known MPEG-7 and Dublin Core standards. Though MPEG-7 allows for semantic representation of multimedia content but lacks the formal explicit semantics and the high-level descriptions.

This work proposes a novel multi-phased top-down approach to develop an aggregator semantic web portal that can serve as a digital marketplace. This portal collects product information from multiple sources and keeps it at a single place. We exploit the Semantic Intelligence Technologies (SITs) in order to achieve our aim. The proposed framework comprises majorly two phases, knowledge modeling (the creation of product ontology; the ingestion and integration of data from product catalogues; the storage of data in triple store) and knowledge consumption (exploration, search and consumption of stored knowledge). We choose the engineering equipment as the domain and provide three service endpoints to the end user; (a) A Rapid Information Explorer (RIE), (b) a SPARQL endpoint, and (c) a smart engineering equipment recommender (EER) as an outcome of this work. The RIE acts as a keyword search engine, where the end user could search and explore the product catalogues. The SPARQL endpoint provides a rich query engine. The EER suggests new products in the market based on the user's input features. We demonstrate that the approach can provide a rapid information retrieval and search across the vendors under one roof just like Trivago does for the hotels. The end user is able to explore the product catalogues in a user friendly manner and can even query the knowledge store using the SPARQL endpoint. All the data and materials are made available at the github repository https://github.com/semintelligence/Product-Discovery.

2 Overview and proposal

With each new generation of computing architectures, we have new applications, workloads, and workflows. The unstructured Big Data coming from the web, Internet of Things devices, and social media generate an unprecedented amount of multimodal data and multimedia content.

2.1 Challenges in product discovery

Nothing comes without challenges. Making the best match between the offers (product catalogues), the retailers making offers, the requested specifications, the customer's preferences, and the intended use of the product comes as the basic problem in product discovery. While looking for the best match in product offerings, the customer faces following problems:

- *Uncertainty (Incomplete Specifications):* On one hand, the buyers are unaware of their specific requirement. They have a vague and very general idea of what they wish to purchase. On the other hand, the seller may miss some important descriptions while listing the items in his product catalogue. This problem of incomplete specifications requires being flexible enough and adjusting whenever constraints change. The uncertainty due to incomplete representations may result in duplicate entries in product knowledge base being created, and makes product matching difficult due to data scarcity on both sides (buyer and seller).
- *Monopoly:* Monopoly comes as a challenge in e-commerce when it is the question of rare and niche products, i.e., custom made products where each product is unique; like leather iPad cases, a unique beaded necklace, and antiques. These are also termed as one-of-a-kind or on-demand products. These types of products are sold by a single vendor and cannot be cross-validated for their category while populating the product knowledge base.
- *Scalability (New Product Offers and Schemes):* When new innovative product offers are introduced, it becomes a challenge to treat new products and categorize them (specifically when they are bringing in new attributes). Schemes such as bundled products and lots of similar products also add a challenge to product categorization. Many different products can be bundled in many possible manners; also similar products can be clubbed in different numbers to make different lots.

– This increases the number of classes multi-fold leaving the solution unscalable.
– It requires retraining the model each time an unseen product offer or scheme is introduced.

- *Products soon Become Outdated:* When existing product offers disappear from the catalogue, or the schemes offered are no longer alive, it again requires retraining the model.
- *Heterogeneity (Non-Standardized Catalogues):* Another major challenge is the global hierarchy of product classes. The product catalogues are not standardized; different catalogues have different descriptions and representations for the same product class. The effective exploitation of the product catalogues requires finding relevant data sources and integrating them by combining the elements.
- *Natural Language Queries:* The user queries may be unambiguously specified. For e.g. "Suggest a laptop with maximum possible memory". It is not clear how much memory is sufficient and whether RAM is queried or Hard Disk.

2.2 Previous key contributions

Knowledge graphs have tremendous potential to exploit the data of web stores for better recommendations. Previous research has depicted the importance of SITs in information systems.

2.2.1 Product categorization and matching

Lots of studies have been focused on structuring the product catalogs, i.e., characterizing things (products, articles, human beings etc.) with categories [28]. Pohorec et al. [36] analyzed and compared different approaches for semantic tagging of machine-readable data on the web. The authors discuss formats available: RDFa (https://rdfa.info/), Microdata (https://en.wikipedia.org/wiki/Microdata_(HTML)) and Microformats (http://microformats.org/). The task has traditionally been addressed in two ways.

1. **Generate Topics from Scratch:** This approach generates categories from the text in a bottom-up style using clustering [5, 16]. One of the first works in this approach to identify categories is the Topic Detection and Tracking (TDT) program developed by DARPA [2] in the domain of broadcasted news. To organize a stream of documents, the TDT program performs cluster analysis. Some methods use keywords and phrases and perform analysis to identify clusters [13, 44]. Yet some other methods utilize word embeddings to quantify semantic similarities in unstructured text [47]. Ristoski et al. 2018 provides a machine learning approach for product categorization and matching.
2. **Leveraging a domain vocabulary or ontology:** This approach uses classifiers that assign a set of pre-existent categories (from controlled vocabularies like schema.org) to the given product description with an advantage to annotate the thing with clean and formally-defined categories.

The first set of approaches tends to produce noisier and less interpretable results [34]. The latter solution relies on a set of well-defined product categories, but requires a good vocabulary in the domain. Ontologies are fundamental to ensure the consistency between datasets since they formally represent and precisely define concepts and their relationships. They improve the semantic content of the data and link datasets at a schema level. This second set of approaches may utilize supervised machine learning, deep learning, or unsupervised machine learning.

2.2.2 Categorization standards

Various categorization standards exist for products and services details, the major among them being:

- the United Nations Standard Products and Services Code (UNSPSC) standard is available in 14+ languages and is currently managed by GS1 US,
- GS1's Global Product's Classification (GPC) offers a universal set of standards for every type of product.
- eOTD is the ECCMA Open Technical Dictionary of unambiguously described concepts.

- European Union's Common Procurement Vocabulary (CPV) is a vocabulary of services to process the procurement and tendering.
- Schema.org: Google, Yahoo, Yandex, and Microsoft are the four major players who have created and are maintaining schema.org as a vocabulary for structured data on the web. This vocabulary can be used with RDFa, microdata, and also JSON-LD. Using Schema.org on meta-modeling level enables integration to the semantic web vocabulary used by many search engines (e.g. Google) and e-commerce sites (e.g. eBay).

These products and services standards have not received adoption in the mainstream due to many issues including the copyright, like for eCl@ss and the licensing fees, like for eCl@ss and UNSPC. They become much more usable when they are converted to richer representations that can be consumed and interpreted in a Linked Data and Semantic Web context. For this, we require to derive RDFS or OWL versions from these standards. Researchers are working on converting these standards into the ontology variants [PCS2OWL, eCl@ssOWL].

- eCl@ss OWL: Hepp, in [17] proposed an OWL ontology vocabulary eClassOWL to enable the representation of catalogues on the semantic web. Brunner et al. in [6] provided an overview of the improvements that are possible for managing product data. However at that time SITs were not mature enough and lacked standards, Brunner presented an information system built on semantic technologies for discovering product data.
- Good Relations ontology: In [18], Hepp presented ontology GoodRelations for data representation of products. It is an OWL Lite ontology derived from the categorization standard eCl@ss 5.1 that covers the representational needs of typical business scenarios in the commodity segment.
- Product Ontology: The product structuring knowledge can be represented in the form of a formal ontology expressed in Web Ontology Language (OWL). Semantic Web Rule Language (SWRL) can extend the OWL for providing capabilities to express constraints that are not expressible as OWL axioms. All product configuration constraints can be expressed as SWRL rules making the solution similar to rule based approaches, allowing to reason on the ontology using inference engines. One such ontology is Product ontology that extends Schema.org and Good Relations vocabularies with links to hundreds of thousands precise definitions for types of product in Wikipedia.

All the above existing ontologies are used to capture the textual metadata of the products. Thus if we want to use multimedia for product discovery specially via images then it is required to capture the image metadata to map with the product so that the user can search the product using image specifications. There are many existing ontologies for image annotation also, the major image metadata capturing ontologies are;

- Image Annotation on the Semantic Web: In 2006, developed a metadata for describing the multimedia contents. The metadata is named as VRA (Visual Resources Association) which uses core data from the Dublin-core such as title, date, creator. This metadata is designed for 4 use case images namely, Scientific Image, Cultural Image, Media and World Images. Ref: https://www.w3.org/TR/swbp-image-annotation/#VraCore
- Dublin-core: Created in 1995 in Dublin that describes the broad and generic elements for resource descriptions which consists of 15 elements in total. Ref: http://dublincore.org/specifications/dublin-core/dcmi-terms/

- EXIF: Exif stands for Exchangeable image file format developed by Kankazi in 2003. It is the vocabulary that defines the exif tags as RDF properties to describe the picture data. Ref: http://www.w3.org/2003/12/exif/ns

The Aletheia architecture proposed in 2010 federates information from heterogeneous sources [43]. [40] presents the most commonly used ontologies for multimedia content. Fitzpatrick et al. in [14] presented a holistic approach for master data management. This work focused on the usage of the same data structures in every phase of the product life cycle. In [41], Stolz et al. presented one more approach focusing on integration. The authors proposed transformation from the BMECat catalogue format to GoodRelations ontology. Nederstigt et al. [31] proposes FLOPPIES for semi-automatic population of ontology from Web stores. Vandic et al. [42] is an example of the use of semantic tagging products in Internet stores. Platform as a source uses its own database that is constantly updated by pinging web shops that use RDFa for semantic labeling. Recent works [30] have shown Microdata format as the most commonly used markup format, along with highest domain coverage. They have also commented that schema.org vocabulary is most frequently used to describe products.

2.2.3 Knowledge graphs

Many organizations have begun bringing Knowledge Graphs in their information ecosystem as they come with such technologies and tools that solve most of the problems they face, like interoperability, findability and more [9, 25]. Knowledge graphs are very close to human thinking as they are built upon the RDF data model, thereby allowing us to capture relationships in the same manner as the human brain processes information. A knowledge graph (KG) is a very large semantic network that represents knowledge without a strict schema by integrating information from variegated heterogeneous sources. A knowledge graph is a network structure that captures the rich knowledge contained by items. It provides abundant reference values for providing recommendations, as it expands the amount of information of each item and strengthens the connections between them. Examples include Wordnet, Google knowledge graph, DBPedia, Yago, WikiData. No clear definition of knowledge graph exists till date and the term is fundamentally used interchangeably with ontology. The most general view calls the complete thing (concepts, relationships, and instances) a knowledge graph that can be queried for instances, and the schema only (i.e., concepts and relationships) an ontology that can be used for inference. Other views exist with disagreements on size, and role of semantics. We will take the general view here and speak of the ontology as the schema / data model and the KG as the actual instance data (like rows in a RDBMS) built in accordance with some ontology (schema).

The classes and relationships are termed as TBox (Terminological Box) and the instances termed as ABox (Assertional Box) in Description Logic; both of them together making up a knowledge graph. TBox statements are used and stored as a schema, while ABox statements are stored as instance data within transactional DBMSs. The NOSQL databases and the RDF databases, TBox statements (data model) and ABox statements (data/instances) are stored using the same approach. In view of all above, KG = ONTOLOGY (TBox) + DATA (ABox). In a knowledge graph, the TBox is not so big but they have a very large ABox. The readers have to make a note that every RDF graph (like the marks in course Semantic Web of students of a university) is not a knowledge graph as it may not capture the required semantics.

Knowledge graphs have demonstrated prospects in all the domains and fields and have been recognized as the core element for semantic representation of knowledge in information systems [10]. The knowledge graphs have been customized for industrial products and services also [26]. Addressing the challenges related to construction of knowledge graphs requires well-founded research, the investigation of concepts, development of tools, and also evaluation parameters and methods. [24] utilizes Formal Concept Analysis (FCA) for constructing Knowledge Graphs. R2RML was recommended in 2012 by W3C, and since then, different extensions, alternatives and implementations were proposed [8]. Certain approaches followed the ETL-like paradigm, e.g., SDM-RDFizer [19] and FunMap [22], while others the query-answering paradigm, e.g., Ontop [7]. Besides R2RML-based extensions, alternatives were proposed, e.g., ShExML [15], as well as methods to perform data transformations while constructing knowledge graphs, e.g., FunUL [23].

2.3 Gaps and scope

The data deluge refers to a large pile of data that is more than dead if left unanalyzed in the absence of efficient search and exploration techniques. Most of the existing techniques to product categorization, matching, search and exploration rely on syntactic approaches, thus ignoring valuable semantic information during the process. These systems are designed using machine learning algorithms that require a big number of labeled data points [27]. Despite all the efforts, we still lack practical unsupervised approaches for classifying entities according to a comprehensive and granular set of categories [1]. The existing works do not properly reflect the specific semantics of the underlying standards. All the mechanisms lack precision when representing the exact context of knowledge. This context is very important for analysis and is better represented in the form of relationships in knowledge graphs [32].

Background knowledge in the form of knowledge graphs can augment the training data of intelligent systems and hence improve both the accuracy and explainability [3]. Mostly, the ontology engineering work is done in academic context only without wide acceptance in industry, therefore the number of practically relevant, quality product ontologies on the Web is still limited [37]. A cost-efficient solution is highly appreciable and recommended that is able to scale up to industry standards and can accommodate business needs on the Web of Data. The direction towards learning automatic semantic relationships on unstructured sequence of e-commerce text has been of less focus. This work proposes the inclusion of knowledge bases information in order to improve searching, matching, and classification tasks of products using their metadata. Integrating the descriptions as well as specifications of different products into a single representation requires organizing all the products across vendors in a single taxonomy.

3 System architecture and framework

In order to overcome the challenges of the e-Commerce marketplace, this work aims at laying the foundation of a search and price comparison semantic web portal for product offers provided by various eShops, OEMs (Original Equipment Manufacturer) and the local Retailers. The portal provides two interfaces: one to the new products from manufacturers/retailers/eShops to enter into the knowledge base, and the other to the end users to get suggestions on which products to purchase. The end user can input raw technical specifications of the product to be purchased and gets the top n recommendations (in descending order of similarity) on the products available via selling resources. Each recommended product is accompanied with the detailed specifications including the vendor, cost, and many more.

What is required is a content management system with semantic intelligence [39] that is more than just a repository of unstructured documents, or data stored in a SQL database, or even NoSQL databases. The most pragmatic approach is to look at the strengths and capabilities that are existent in our own data and leverage them by practices like metadata, taxonomies, ontologies, and knowledge graphs. The recommendation engine will be powered by the offbeat semantic intelligence technologies and rich metadata to replace the manual process and allow for advanced Artificial Intelligence capabilities and high quality and targeted recommendations. Figure 1 presents the architecture of the smart Engineering Equipment Recommender (EER) for facilitating recommendations utilizing the web semantics. The proposed system architecture comprises majorly two phases / modules and the underlying knowledge base. The knowledge base is an ontology-based representation of domain knowledge with all the required concepts and schemes available in the Terminology Box or TBOX and all the known individuals available in the Assertion Box or ABOX.

3.1 Knowledge modeling

The system admin generates a taxonomy and ontology that is comprehensive, user friendly, granular, and applicable to multiple use cases. The content, i.e., the product instances are extracted from different sources (structured or unstructured) and the knowledge graph is created in accordance with the developed ontology. The knowledge of product offers is to be represented in the form of semantically enriched data and then hosted. To describe the data of product catalogues, metadata is added by using a self-curated vocabulary. RDF has already established itself as a standard for expressing the semantics of MPEG-7 and video metadata descriptions.

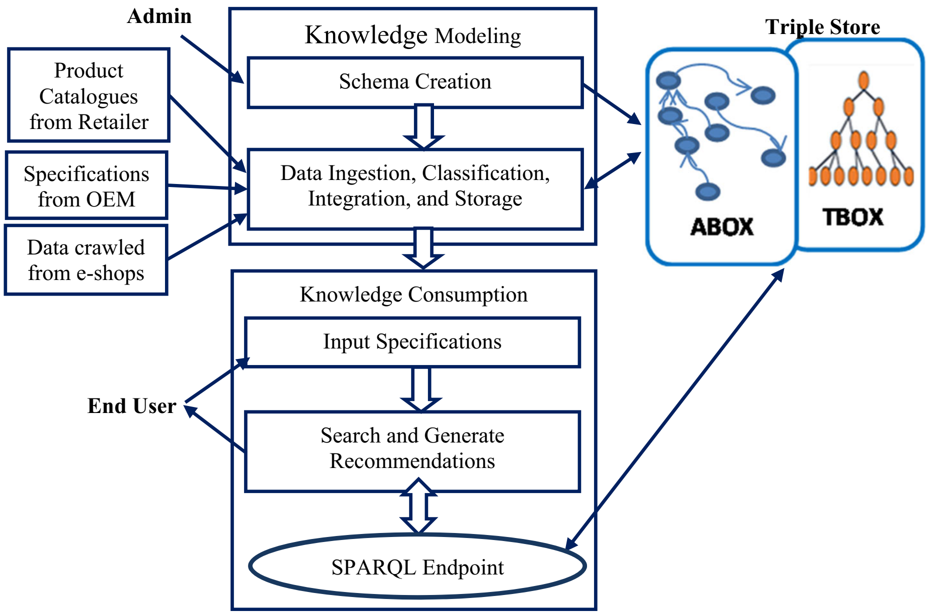

Admin
Knowledge Modeling
Schema Creation
Product Catalogues from Retailer
Specifications from OEM
Data crawled from e-shops
Data Ingestion, Classification, Integration, and Storage
Triple Store
ABOX
TBOX
Knowledge Consumption
Input Specifications
End User
Search and Generate Recommendations
SPARQL Endpoint

Fig. 1 Product discovery framework

3.1.1 Schema creation

To provide recommendations over the input specifications, the product information (either in catalogues or on web) from the retailer need to be machine understandable, hence required to be annotated semantically. Different forms of metadata and pre-defined vocabularies are useful for semantic annotation of content. The information that expresses information about the meaning and context of some data is ***metadata.*** Metadata helps in fetching all related content about a data precisely. Because of the limited support by people creating websites and a very slow and non-existent adoption of semantic technologies, some technologies started combining semantic information along with the already existing content by increasingly adopting Microformats, RDFa, and Microdata as semantic markup languages [29]. These approaches introduce ambiguity as the same thing can be annotated in different ways by different people. To normalize the terms for providing good metadata, a pre-existing taxonomy is required. Either use some existing taxonomy, or author one. After authoring the taxonomy, it is to be published on the LOD cloud. ***Taxonomies and Thesauri*** are very closely related controlled vocabularies to organize relevant data by describing relations between concepts and their labels including synonyms. They allow for effective classification and categorization of both structured and unstructured information for the purposes of findability and discoverability. ***Ontologies*** are generalized semantic data models that define the vocabulary of a domain and the relationships between the concepts of vocabulary. Ontologies standardize the terminology of an application domain, and present a unified view of data sources, hence making information sharing and data integration easy.

Metadata, Taxonomies, and ontologies together bring structure to the unstructured data and provide information to the machines in a machine-readable format, so that meaningful inference can be drawn. We have decided to leverage ontologies for knowledge graphs schema. In the knowledge graph building efforts, the first question that arises is the choice of vocabulary to use. We have built an ontology-based representation of the product catalogue in a manner that it is interoperable and can be combined with offers from other retailers. We call it the engineering equipment ontology (EEO). In this knowledge modeling task, we formalize the catalogue by representing the product descriptions as ontological classes in Web Ontology Language (OWL). To integrate the multimedia aspect in the product discovery, we are using the EXIF vocabulary to map the product image metadata with the product metadata that will allow users to search the product using image data also. The EEO will serve as the meta-knowledge for the products and/or services and will be updated periodically for any new description or product type that comes in.

We identified the domain of discourse's terminologies for concepts or classes, relationships, and attributes. The relationships and attributes both can be considered as properties. Relationship between two classes is termed as object property, and attribute of a class is termed as data property. Some properties of the Product class might be: *hasModelNumber, hasBrand, hasMemory, hasDimensions, hasPrice and so on.*

In the "Web of Data" model, "Thing" is the top-class entity as an abstraction of any physical or virtual entity. It is the super class of all the entities. We create sense out of input data by annotating our inputted data set with universal known concepts available on the linked open data cloud. In the linked data, the concept, for example, gr:ProductOrServiceModel is already recognized uniquely through IRI whose meaning is interoperable across domains and hence comprehensive not only to humans but also to machines. Because of open world assumption (OWA) in ontologies, we have captured possible properties that could apply to many, but not necessarily all products. The EEO ontology will serve as a general data model and a reusable framework that we will use to describe the instances as we discover them. Figure 2 depicts the graphical view of Engineering Equipment Ontology (EEO) and a very small portion of the Engineering Equipment Knowledge Graph (EEKG).

In the above statements, eeo, gr, schema, and dc are the namespaces for Engineering Equipment, GoodRelations, schema.org, and Dublin core vocabularies, respectively. All these namespaces describe the core taxonomy related to products like productionNumber, manufacturingDate, license and so on. If the user wants to search for any product, then the particular product specifications are required rather than the product making details. Thus, these ontologies lack the specification taxonomy which is required to capture the whole details of the product and compare the same with two or more products based on its specifications.

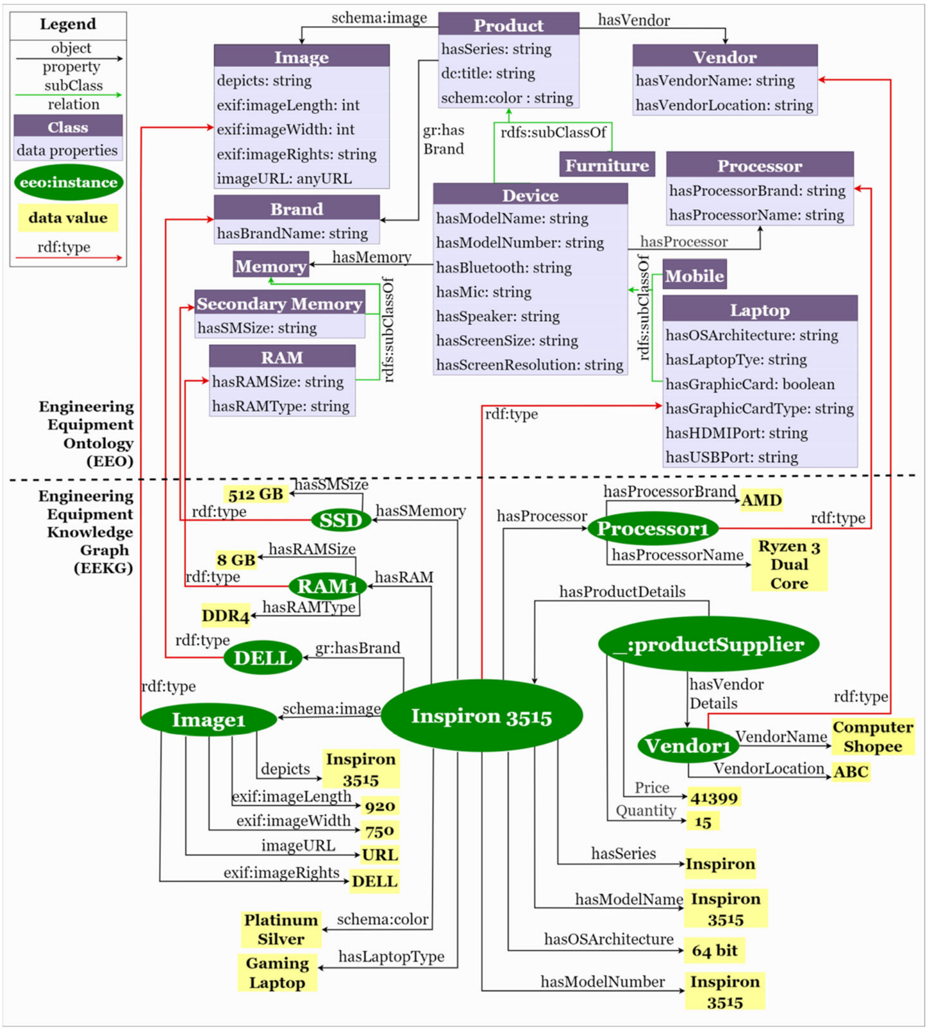


Fig. 2 Engineering Equipment Ontology (EEO) and Knowledge Graph (EEKG)

From these namespaces, we are reusing some of the concepts or properties mentioned in the standard vocabularies. For image metadata we are using all the properties from EXIF vocabulary. Some of the common properties are *length, width, model, copyright*. Some properties which the existing image metadata does not address are included in EEO such as *frontView, frontViewColor, rearView and rearViewColor*. This enables EEO to have the possible level of reusability using IRI of concepts and hence preserving machine understanding for the concept.

3.1.2 Data ingestion

When the graph model, EEO, is ready, we start applying it to the product catalogue data inputs, and in effect populate the graph with data, i.e., instances of vended products. There are two types of data sources, structured and unstructured. In cases when the data source is structured like a relational database, or the data is highly dynamic; we prefer to keep the data in-place and utilize virtual graphs as a mapping from SPARQL to SQL. This strategy is called the Virtualization approach [Poggy et al. 2008] and maintains the freshness of data as it is always live and avoids any duplication, but somehow degrades performance while querying. In cases when the data source is unstructured, or less dynamic; we should utilize the semantic ETL (Extract, Transform, Load) pipeline to extract data from the source periodically, transform it into RDF, perform semantic tagging using ontology, and load into the graph. This strategy is called the Materialization Engine. As product catalogues are highly unstructured and mostly static about product offers, except the newly introduced products, we take the ETL approach. Currently we have used the dummy data for populating the knowledge base using the EEO taxonomy. The data is collected for various products from 3 types of vendors viz. OEMs, eShops, and Retailers. For giving the system a working glimpse, we are using one of the product categories i.e. Laptop as an example. Same product can be available from different vendors with differences in quantity and prices. We call the resultant knowledge graph the Engineering Equipment Knowledge Graph (EEKG). The extraction, transformation, and loading of data are explained in detail in the following subsections.

1. **Extraction:** The extraction process had been carried out by scrapping the products information over the internet. The scraping was done using python based open source libraries -

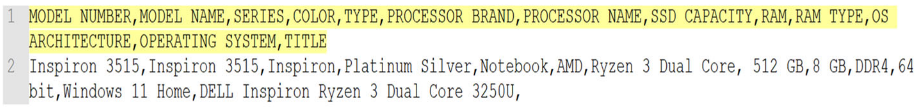
i) requests - https://requests.readthedocs.io/en/latest/
ii) bs4 - https://www.crummy.com/software/BeautifulSoup/bs4/doc/

The requests library first sends requests to the server to obtain the required html page. Once the page is received from the server, bs4 library (BeautifulSoup4) is then used to crawl the data from the requested html page. The extracted information is then converted into CSV (comma separated values) format. This is the data we will work upon and the same can be downloaded from https://github.com/semintelligence/Product-Discovery. The CSV data contains 50 attributes containing product details. For the sake of simplicity, we are taking the following 14 attributes of products throughout the writing:

Model Number, Series, Model Name, Color, Type, Processor Brand, Processor Name, SSD Capacity, RAM, RAM Type, OS Architecture, Operating System, Brand, and Image

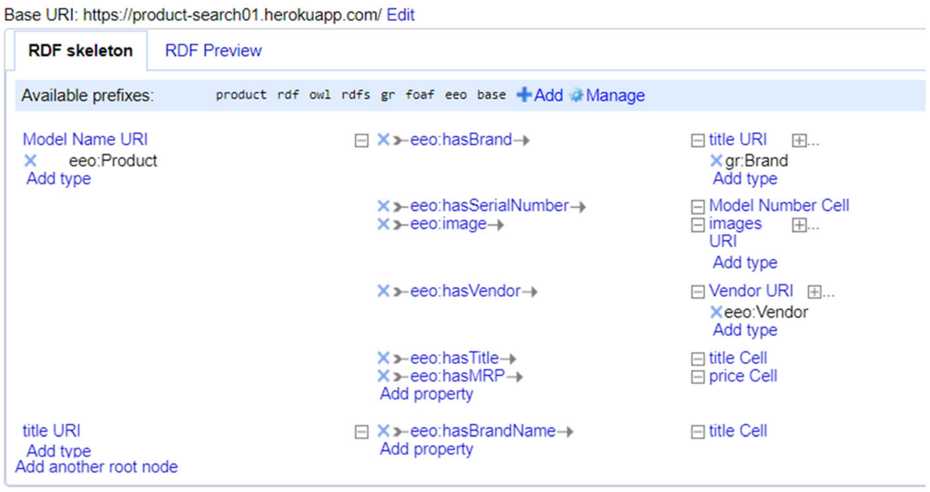
2. **Transformation:** Many tools and techniques have been proposed in literature for converting data from any format into linked data in RDF format [12]. We transform the extracted product instances into semantically structured data in RDF serialization aligned with the Engineering Equipment Ontology (EEO). Before we could do any knowledge representation of the inputted data, we had to do data pre-processing activities like data wrangling, cleansing etc. For this, we used an open source tool called OpenRefine (https://openrefine.org/) and for defining RDF definition on the data set, RDF Extension (https://github.com/stkenny/grefine-rdf-extension), an extension of OpenRefine, has been used. The RDFExtension of OpenRefine allows semantically curating the inputted data with some ontology. Here we used EEO as a domain knowledge representation framework for semantic annotation of inputted data. The developed EEO has conceptualized the domain knowledge and at the same time referring and hence reusing the knowledge from the standard vocabularies. With RDFExtension, we can provide a mechanism to provide axioms and assertions for semantically annotating the primary key of a row, here Model Number for example, by making its values an individual node of concept from gr:ProductOrServiceModel. We then annotate each node with the available data from other columns, with the properties connecting with another node or data.

Consider the dataset extracted in the last section to be converted into the initial EEKG during bootstrapping. Figure 3a is a partial screenshot of the dataset and Fig. 3b depicts the schema alignment in OpenRefine for the row shown in the dataset. Figure 2 depicts the graphical view of the same product "Inspiron 3515" along with the information that it is being sold by the vendor "Computer Shoppe" with the available quantity in stock as 15 and Price as 41,399 units.



```
MODEL NUMBER,MODEL NAME,SERIES,COLOR,TYPE,PROCESSOR BRAND,PROCESSOR NAME,SSD CAPACITY,RAM,RAM TYPE,OS
ARCHITECTURE,OPERATING SYSTEM,TITLE
Inspiron 3515,Inspiron 3515,Inspiron,Platinum Silver,Notebook,AMD,Ryzen 3 Dual Core, 512 GB,8 GB,DDR4,64
bit,Windows 11 Home,DELL Inspiron Ryzen 3 Dual Core 3250U,
```

a



Base URI: https://product-search01.herokuapp.com/ Edit

RDF skeleton | RDF Preview

Available prefixes: product rdf owl rdfs gr foaf eeo base +Add Manage

```
Model Name URI     ⊟ ✕ >-eeo:hasBrand->            ⊟ title URI  ⊞...
✕   eeo:Product                                      ✕gr:Brand
Add type                                             Add type
                     ✕ >-eeo:hasSerialNumber->     ⊟ Model Number Cell
                     ✕ >-eeo:image->               ⊟ images  ⊞...
                                                     URI
                                                     Add type
                     ✕ >-eeo:hasVendor->           ⊟ Vendor URI  ⊞...
                                                     ✕eeo:Vendor
                                                     Add type
                     ✕ >-eeo:hasTitle->            ⊟ title Cell
                     ✕ >-eeo:hasMRP->              ⊟ price Cell
                     Add property
title URI          ⊟ ✕ >-eeo:hasBrandName->        ⊟ title Cell
 Add type            Add property
Add another root node
```

b

Fig. 3 **a** Partial Screenshot of the dataset **b** OpenRefine Transformation Mechanism

3. **Loading (knowledge hosting):** RDF is more expressive than any other language to model complex and unstructured data. Because of the structure of knowledge graphs, they should be backed by an RDF data store in order to exploit the embedded context. Two different approaches are utilized for storing this semantically annotated data. If this data is meant to be exploited by some use case in a manner to be understandable by the automated agents, for say SEO then it is stored in a JSON-based document database, e.g., MongoDB. In this case, the data is queried over an API and no RDF reasoning is possible. If the data is meant to be exploited by the enterprises for reasoning, it is to be stored in a triple store and querying is simplified through SPARQL endpoint

As we have to use our data in our portal itself for recommendation, we take the second approach of storing in a triple store. Triple stores serve the road ahead from Big Data to Smart Data. The RDF triple stores are based on W3C standards and they store triples (subject-predicate-object). Triple stores have a lot of foreseen benefits for storing RDF data. As all the triple stores speak the same language, moving data between stores is easier. They are queried over existing HTTP, so no bridging solutions are required. They allow multiple values for a variable thus facilitating multiple internationalized languages. For triple stores, there is a choice between triple stores built from ground up, i.e., RDF triples (like Jena TDB, GraphDB), also called native triple stores; and the triple stores that are based on relational databases and simply provide a SPARQL interface (like Virtuoso, Jena SDB), also called non-native triple stores. Amongst the open-source implementation of triple stores, we had a choice between Jena TDB, OpenLink Virtuoso, Blazegraph, RDFox, Sesame (rdf4j), Stardog. The internal RDF graph structure can handle the scale, speed, and variety of data and is flexible enough not to re-defined its schema every time when new data is added, therefore, we go for the native triple store Jena TDB by Apache Foundation.

Figure 4 describes our data hosting architecture for the Product Discovery web application. The figure can be read from left to right where the git repository is provided by the heroku web hosting service to load the RDF data and triple store configurations (Jena Fuseki Server in our case) via a Command Line Interface (CLI). The SPARQL endpoint is then generated automatically, which is used in Django WSGI (Web Server Gateway Interface) to connect the data with the front end of the application.

3.2 Knowledge consumption

Now all the required knowledge (schema as well as data) is available in a machine understandable format in the triple store; it is time to query it and fetch valuable insights for various

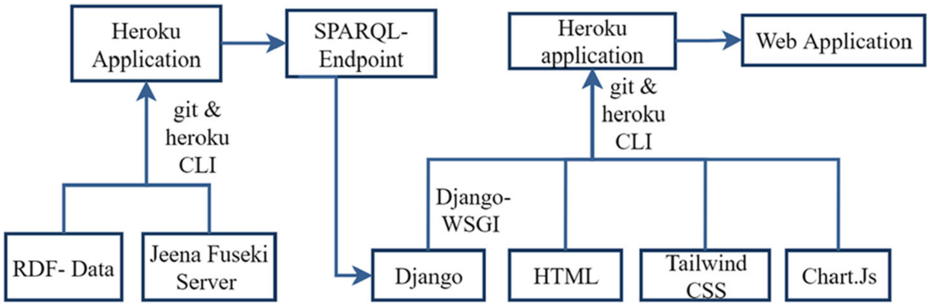


Fig. 4 Heroku data hosting architecture

use cases as resource manager, semantic search, knowledge analytics, question answering, and more. The aggregator semantic web portal exposes one end point for the vendor and three end points for the end user.

(1) **The Vendor EndPoint**: The Vendor Endpoint acts as the input gate for the new products in the market. The vendors can edit the price or quantity in stock of existing products, and also delete some products that are no longer available with them to sell.
(2) **Rapid Information Explorer (RIE):** The RIE acts as a keyword search engine, where the end user could search and explore the product catalogues. The end user is able to explore the product catalogues in a user-friendly manner.
(3) **SPARQL Endpoint:** The SPARQL endpoint provides a rich query engine. It works on an assumption that the user is acquainted with the SPARQL (SPARQL Protocol and RDF Query Language) query writing mechanism.
(4) **Smart Engineering Equipment Recommender (EER):** The EER talks about generating recommendations of top n matched offers in the triple store. The user requests for certain products are matched against the available product categories to generate a knowledge graph. This process of categorization of input specifications iterates over a period of time by suggesting the user to refine the specifications and in turn rearranging the knowledge graph. After the knowledge graph is in place, it is matched with the available product instances and top n product offers are displayed to the user along with the attributes. For instance; as depicted in Fig. 5, Product B is the best choice with make: Apple, price: less than Rs. 1,60,000, and color: Black as desired. Product A matches two features, and product C matches only one.

4 Experiments and evaluation

The proposed approach facilitates operations over aggregated data toward automating the tasks of e-business. It assists in smart analytics by providing automatic indexing and semantically enhancing the metadata of engineering equipment. EER is capable of detecting trends in design of engineering equipment for better understanding. The idea is to provide information and tools to help the customer find the best product across various eShops, in the same manner as Trivago suggests the best hotel deal according to the customer requirements. Through the

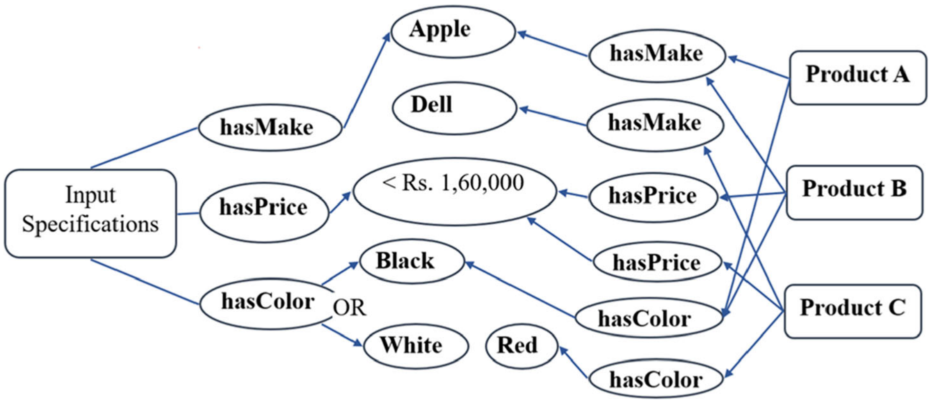


Fig. 5 Product matching

EER, the end user can navigate different available products based upon their specifications, query the knowledge base according to requirements, and additionally can compare prices of the same product with the other vendors (may be OEM or retailer).

We have developed a web application as a Minimum Viable Product (MVP) using Django (https://docs.djangoproject.com/en/4.1/) that is a python (https://docs.python.org/3/) based web development framework. This application can be accessed through exposed endpoints in the form of URLs. The development service is a cloud-native application, which is hosted on a cloud application platform called Heroku (https://www.heroku.com/home). The application has been implemented using the Apache Jena Framework (https://jena.apache.org/), which is used for building a SPARQL endpoint for data consumption.

Refer to Fig. 4 showing how the front-end designing files and Django framework code are committed to the Heroku application via git repository and the Web Application URI is generated. This URI can now be used to access the web application. The final result is a web application that uses linked data as a data source and queries products based on it. The web application can be accessed at https://product-search01.herokuapp.com/. The namespace prefixes used throughout the SPARQL (SPARQL Protocol and RDF Query Language) query are shown in Fig. 6 below:

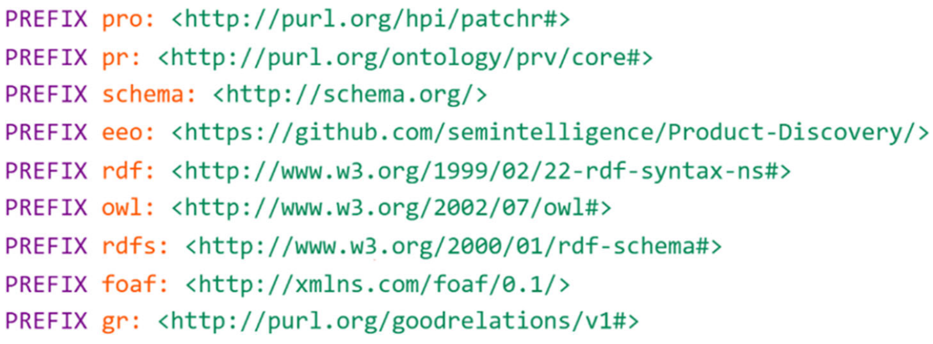
4.1 Evaluating the vendor endpoint

The data coming from vendors can have heterogeneous formats and the chosen data model, i.e., RDF enables effective data integration from multiple sources without enforcing structure on it. As the data is detached from the schema, it is allowed to have unlimited relationships. Data can be inserted, updated, and deleted without changing the data instances.

Once the EEKG has been bootstrapped with the initial knowledge during its birth, all the three interfaces of the end user are up and running. The vendor is provided with the endpoint where all the products will be displayed to him that he sells. This interface allows 3 operations i.e. insert new products, update the quantity/price of existing products, or delete some product that he has now discontinued to sell.

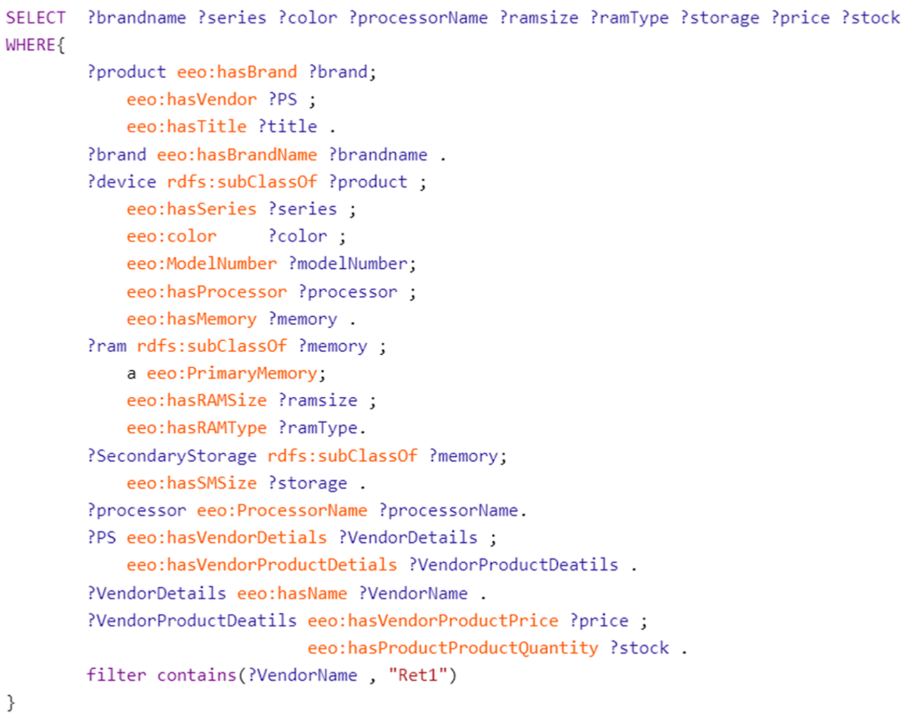
4.1.1 Display all the products vended by the vendor

After successful login, a vendor is displayed the list of all product instances that he has already exposed through this aggregator site. Figure 7a shows the SPARQL query and Fig. 7b shows the output.

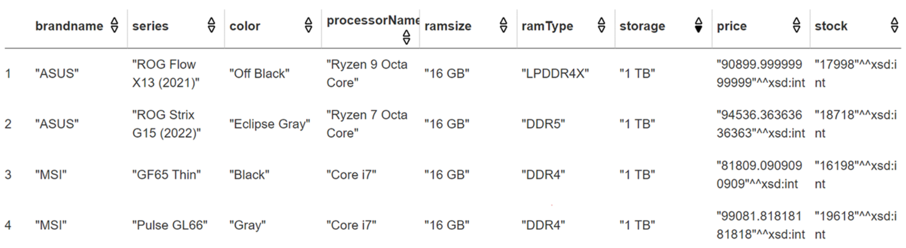


```
PREFIX pro: <http://purl.org/hpi/patchr#>
PREFIX pr: <http://purl.org/ontology/prv/core#>
PREFIX schema: <http://schema.org/>
PREFIX eeo: <https://github.com/semintelligence/Product-Discovery/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX gr: <http://purl.org/goodrelations/v1#>
```

Fig. 6 Prefix namespace list used in the queries

a 

```
SELECT  ?brandname ?series ?color ?processorName ?ramsize ?ramType ?storage ?price ?stock
WHERE{
        ?product eeo:hasBrand ?brand;
            eeo:hasVendor ?PS ;
            eeo:hasTitle ?title .
        ?brand eeo:hasBrandName ?brandname .
        ?device rdfs:subClassOf ?product ;
            eeo:hasSeries ?series ;
            eeo:color     ?color ;
            eeo:ModelNumber ?modelNumber;
            eeo:hasProcessor ?processor ;
            eeo:hasMemory ?memory .
        ?ram rdfs:subClassOf ?memory ;
            a eeo:PrimaryMemory;
            eeo:hasRAMSize ?ramsize ;
            eeo:hasRAMType ?ramType.
        ?SecondaryStorage rdfs:subClassOf ?memory;
            eeo:hasSMSize ?storage .
        ?processor eeo:ProcessorName ?processorName.
        ?PS eeo:hasVendorDetials ?VendorDetails ;
            eeo:hasVendorProductDetials ?VendorProductDeatils .
        ?VendorDetails eeo:hasName ?VendorName .
        ?VendorProductDeatils eeo:hasVendorProductPrice ?price ;
                                  eeo:hasProductProductQuantity ?stock .
        filter contains(?VendorName , "Ret1")
}
```

b 

	brandname	series	color	processorName	ramsize	ramType	storage	price	stock
1	"ASUS"	"ROG Flow X13 (2021)"	"Off Black"	"Ryzen 9 Octa Core"	"16 GB"	"LPDDR4X"	"1 TB"	"90899.999999 99999"^^xsd:int	"17998"^^xsd:int
2	"ASUS"	"ROG Strix G15 (2022)"	"Eclipse Gray"	"Ryzen 7 Octa Core"	"16 GB"	"DDR5"	"1 TB"	"94536.363636 36363"^^xsd:int	"18718"^^xsd:int
3	"MSI"	"GF65 Thin"	"Black"	"Core i7"	"16 GB"	"DDR4"	"1 TB"	"81809.090909 0909"^^xsd:int	"16198"^^xsd:int
4	"MSI"	"Pulse GL66"	"Gray"	"Core i7"	"16 GB"	"DDR4"	"1 TB"	"99081.818181 81818"^^xsd:int	"19618"^^xsd:int

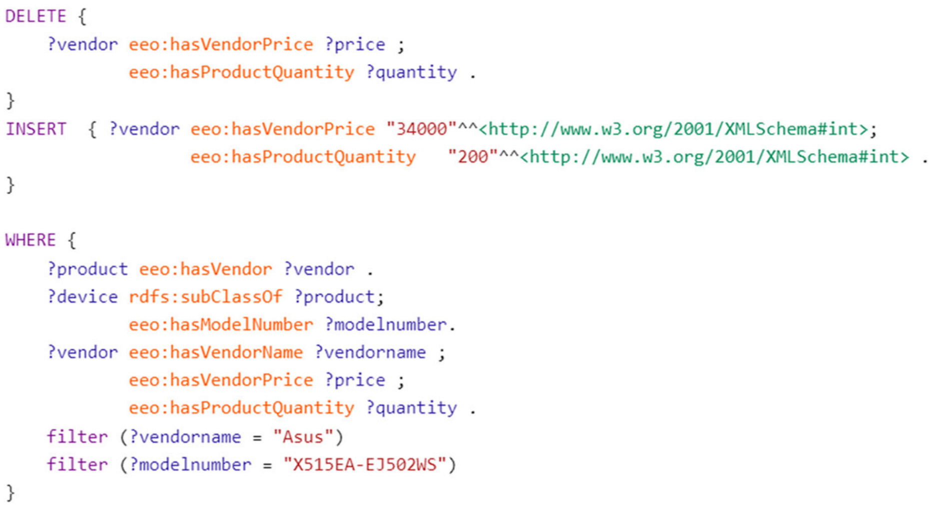
Fig. 7 **a** List of Products Vended by a Specific Vendor - SPARQL QUERY **b** List of Products Vended by a Specific Vendor - OUTPUT

Query parameter Vendor login credentials; It fetches all the vended products of the logged in vendor.

4.1.2 Update the price and quantity

The vendor can select some product whose price and/or quantity he wishes to update. The parameters model number and vendor name will be taken into account to update the required product data by the vendor. Figure 8 shows the SPARQL query.

Query Parameters: Vendor name and Model Number



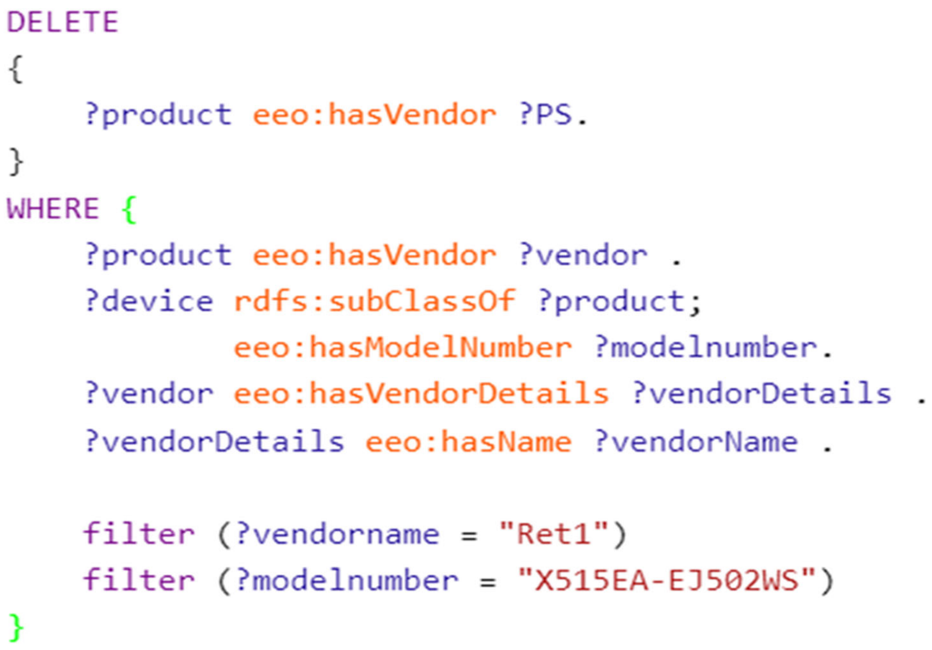
```
DELETE {
    ?vendor eeo:hasVendorPrice ?price ;
            eeo:hasProductQuantity ?quantity .
}
INSERT  { ?vendor eeo:hasVendorPrice "34000"^^<http://www.w3.org/2001/XMLSchema#int>;
                  eeo:hasProductQuantity   "200"^^<http://www.w3.org/2001/XMLSchema#int> .
}

WHERE {
    ?product eeo:hasVendor ?vendor .
    ?device rdfs:subClassOf ?product;
            eeo:hasModelNumber ?modelnumber.
    ?vendor eeo:hasVendorName ?vendorname ;
            eeo:hasVendorPrice ?price ;
            eeo:hasProductQuantity ?quantity .
    filter (?vendorname = "Asus")
    filter (?modelnumber = "X515EA-EJ502WS")
}
```

Fig. 8 Update price and quantity by Vendor

4.1.3 Product discontinuation by the vendor

The vendor can delete any product which he desires to discontinue for selling. The parameters model number and vendor name will be taken into account to delete the required product data by the vendor. Figure 9 shows the SPARQL query.



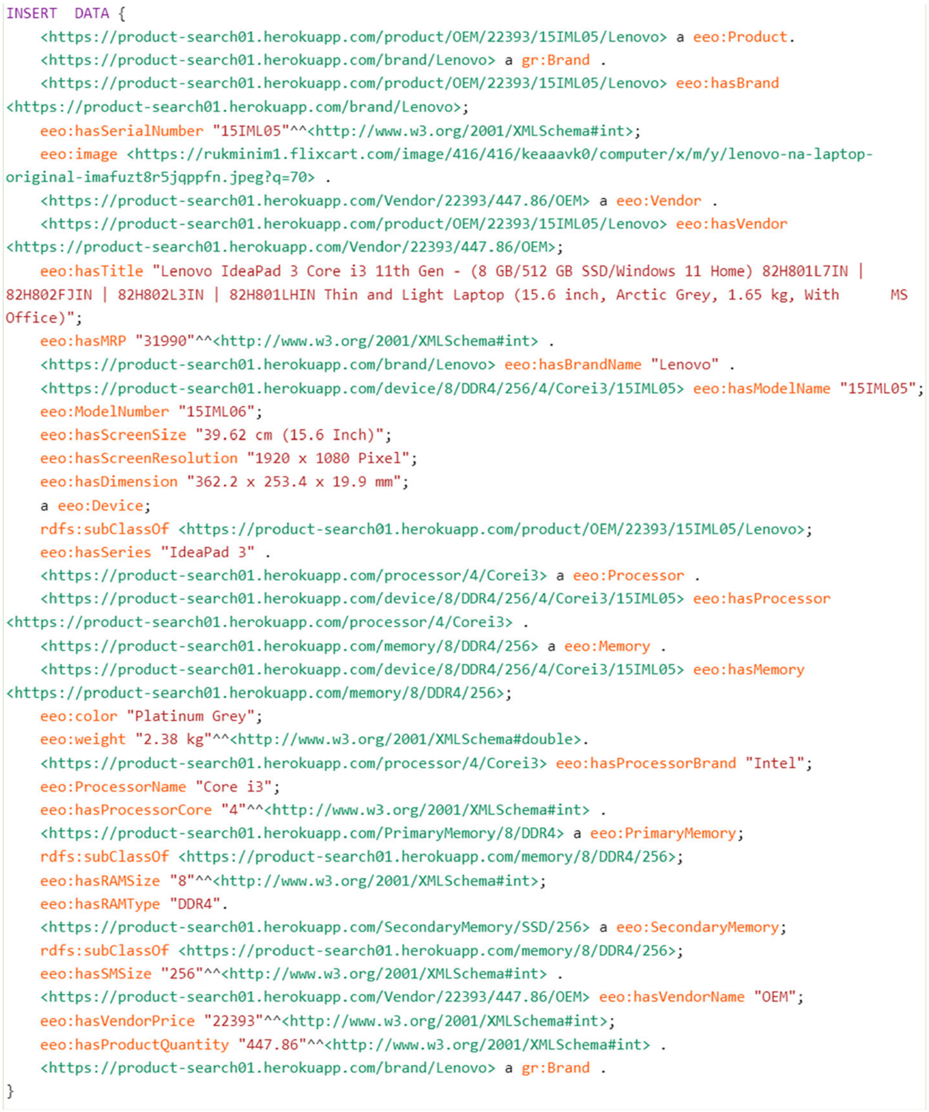
```
DELETE
{
    ?product eeo:hasVendor ?PS.
}
WHERE {
    ?product eeo:hasVendor ?vendor .
    ?device rdfs:subClassOf ?product;
            eeo:hasModelNumber ?modelnumber.
    ?vendor eeo:hasVendorDetails ?vendorDetails .
    ?vendorDetails eeo:hasName ?vendorName .

    filter (?vendorname = "Ret1")
    filter (?modelnumber = "X515EA-EJ502WS")
}
```

Fig. 9 Product discontinuation by a Vendor

4.1.4 Add new product

The vendor may start selling some product that he was not selling before. He is required to input the product specifications into the system. If the product is already available into the knowledge base (being supplied by some other vendor), then only the vendor's price and vendor's quantity will be added to that available product; otherwise complete specifications of the new product also gets added. To check the availability of the product in the knowledge base, the model number of the product is taken into account. Figure 10 shows the SPARQL query.



```
INSERT  DATA {
    <https://product-search01.herokuapp.com/product/OEM/22393/15IML05/Lenovo> a eeo:Product.
    <https://product-search01.herokuapp.com/brand/Lenovo> a gr:Brand .
    <https://product-search01.herokuapp.com/product/OEM/22393/15IML05/Lenovo> eeo:hasBrand
<https://product-search01.herokuapp.com/brand/Lenovo>;
    eeo:hasSerialNumber "15IML05"^^<http://www.w3.org/2001/XMLSchema#int>;
    eeo:image <https://rukminim1.flixcart.com/image/416/416/keaaavk0/computer/x/m/y/lenovo-na-laptop-
original-imafuzt8r5jqppfn.jpeg?q=70> .
    <https://product-search01.herokuapp.com/Vendor/22393/447.86/OEM> a eeo:Vendor .
    <https://product-search01.herokuapp.com/product/OEM/22393/15IML05/Lenovo> eeo:hasVendor
<https://product-search01.herokuapp.com/Vendor/22393/447.86/OEM>;
    eeo:hasTitle "Lenovo IdeaPad 3 Core i3 11th Gen - (8 GB/512 GB SSD/Windows 11 Home) 82H801L7IN |
82H802FJIN | 82H802L3IN | 82H801LHIN Thin and Light Laptop (15.6 inch, Arctic Grey, 1.65 kg, With      MS
Office)";
    eeo:hasMRP "31990"^^<http://www.w3.org/2001/XMLSchema#int> .
    <https://product-search01.herokuapp.com/brand/Lenovo> eeo:hasBrandName "Lenovo" .
    <https://product-search01.herokuapp.com/device/8/DDR4/256/4/Corei3/15IML05> eeo:hasModelName "15IML05";
    eeo:ModelNumber "15IML06";
    eeo:hasScreenSize "39.62 cm (15.6 Inch)";
    eeo:hasScreenResolution "1920 x 1080 Pixel";
    eeo:hasDimension "362.2 x 253.4 x 19.9 mm";
    a eeo:Device;
    rdfs:subClassOf <https://product-search01.herokuapp.com/product/OEM/22393/15IML05/Lenovo>;
    eeo:hasSeries "IdeaPad 3" .
    <https://product-search01.herokuapp.com/processor/4/Corei3> a eeo:Processor .
    <https://product-search01.herokuapp.com/device/8/DDR4/256/4/Corei3/15IML05> eeo:hasProcessor
<https://product-search01.herokuapp.com/processor/4/Corei3> .
    <https://product-search01.herokuapp.com/memory/8/DDR4/256> a eeo:Memory .
    <https://product-search01.herokuapp.com/device/8/DDR4/256/4/Corei3/15IML05> eeo:hasMemory
<https://product-search01.herokuapp.com/memory/8/DDR4/256>;
    eeo:color "Platinum Grey";
    eeo:weight "2.38 kg"^^<http://www.w3.org/2001/XMLSchema#double>.
    <https://product-search01.herokuapp.com/processor/4/Corei3> eeo:hasProcessorBrand "Intel";
    eeo:ProcessorName "Core i3";
    eeo:hasProcessorCore "4"^^<http://www.w3.org/2001/XMLSchema#int> .
    <https://product-search01.herokuapp.com/PrimaryMemory/8/DDR4> a eeo:PrimaryMemory;
    rdfs:subClassOf <https://product-search01.herokuapp.com/memory/8/DDR4/256>;
    eeo:hasRAMSize "8"^^<http://www.w3.org/2001/XMLSchema#int>;
    eeo:hasRAMType "DDR4".
    <https://product-search01.herokuapp.com/SecondaryMemory/SSD/256> a eeo:SecondaryMemory;
    rdfs:subClassOf <https://product-search01.herokuapp.com/memory/8/DDR4/256>;
    eeo:hasSMSize "256"^^<http://www.w3.org/2001/XMLSchema#int> .
    <https://product-search01.herokuapp.com/Vendor/22393/447.86/OEM> eeo:hasVendorName "OEM";
    eeo:hasVendorPrice "22393"^^<http://www.w3.org/2001/XMLSchema#int>;
    eeo:hasProductQuantity "447.86"^^<http://www.w3.org/2001/XMLSchema#int> .
    <https://product-search01.herokuapp.com/brand/Lenovo> a gr:Brand .
}
```

Fig. 10 Add new product by OEM (Lenovo)

4.2 Rapid information explorer (RIE)

HTTP Endpoint: ***Product Search (product-search01.herokuapp.com).***

Query Parameter: On passing any keyword of the domain, it fetches all possible information related to the inputted keyword. Figure 11 depicts the code snippet for the same.

4.3 SPARQL endpoint

HTTP Endpoint: ***Document (product-search01.herokuapp.com).***

Figure 12 shows the SPARQL Endpoint.

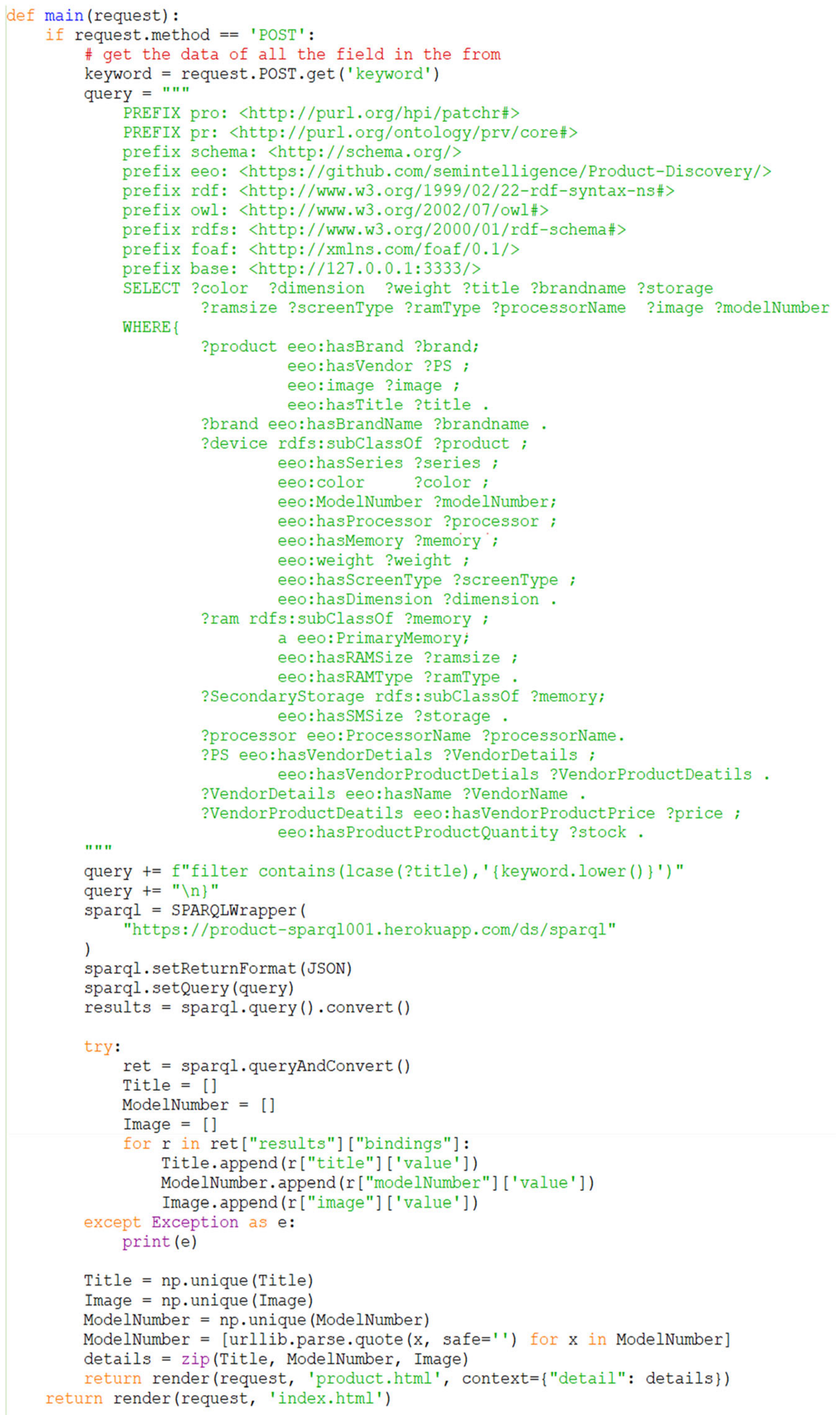
4.4 Engineering equipment recommender (EER)

Scenario User inputs more than one specification of the product into the EER. The relevant products matching all the specifications will be provided as the output. Figure 13a shows the SPARQL query and Fig. 13b shows the output.

5 Discussion

This project is being developed as a not-for-profit research to benefit the end users who are struggling in understanding the specific products that meet their requirements. The system will be made available through a semantic web portal that will be a product search aggregator website. With the time to come, there will soon be no company site but all aggregator sites. Even in Google search the first and second place is taken by the aggregator sites. This multi-phased top-down approach to a digital marketplace provides tailored product recommendations to connect customers with the exact product/service of their choice. The approach followed provides the following specific performance benefits. Our work is different from the existing ontology-based product matching and categorization proposals available in many respects.

- **Data Representation and Storage:** Rather than using the relational database representation for storing the data, the system uses the graph representation of data. In the e-commerce domain, due to the rise of more products and their categories each with a variety of offers and specifications from different manufactures and online retailers, it becomes very difficult to maintain the data in the strict rigid structure like the relational databases or nosql databases. Such data representation requires data to be stored in multiple tabular formats and querying such databases for the recommendation and analysis purpose becomes complex to complex queries. Knowledge graphs overcome this problem with the flexibility structure that helps to store the data with nodes and edges which makes it scalable for usage. This property is hard to find in other databases. As the data is represented in the form of a graph it reduces the data redundancy, loss of data if any data to be removed, and easy insertions. Such representation is very useful for the recommendation system where similar kinds of data need to be presented to the user from various vendors with different pricing for the same product. It also helps to understand the retrieval of the data by capturing the user search behavior.

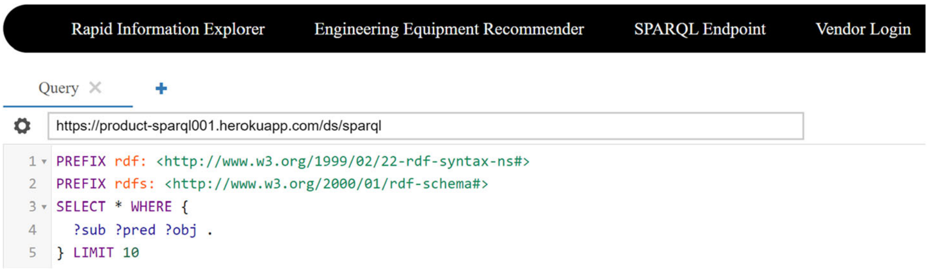


```
def main(request):
    if request.method == 'POST':
        # get the data of all the field in the from
        keyword = request.POST.get('keyword')
        query = """
            PREFIX pro: <http://purl.org/hpi/patchr#>
            PREFIX pr: <http://purl.org/ontology/prv/core#>
            prefix schema: <http://schema.org/>
            prefix eeo: <https://github.com/semintelligence/Product-Discovery/>
            prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
            prefix owl: <http://www.w3.org/2002/07/owl#>
            prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
            prefix foaf: <http://xmlns.com/foaf/0.1/>
            prefix base: <http://127.0.0.1:3333/>
            SELECT ?color  ?dimension  ?weight ?title ?brandname ?storage
                    ?ramsize ?screenType ?ramType ?processorName  ?image ?modelNumber
            WHERE{
                    ?product eeo:hasBrand ?brand;
                             eeo:hasVendor ?PS ;
                             eeo:image ?image ;
                             eeo:hasTitle ?title .
                    ?brand eeo:hasBrandName ?brandname .
                    ?device rdfs:subClassOf ?product ;
                            eeo:hasSeries ?series ;
                            eeo:color     ?color ;
                            eeo:ModelNumber ?modelNumber;
                            eeo:hasProcessor ?processor ;
                            eeo:hasMemory ?memory ;
                            eeo:weight ?weight ;
                            eeo:hasScreenType ?screenType ;
                            eeo:hasDimension ?dimension .
                    ?ram rdfs:subClassOf ?memory ;
                            a eeo:PrimaryMemory;
                            eeo:hasRAMSize ?ramsize ;
                            eeo:hasRAMType ?ramType .
                    ?SecondaryStorage rdfs:subClassOf ?memory;
                            eeo:hasSMSize ?storage .
                    ?processor eeo:ProcessorName ?processorName.
                    ?PS eeo:hasVendorDetials ?VendorDetails ;
                            eeo:hasVendorProductDetials ?VendorProductDeatils .
                    ?VendorDetails eeo:hasName ?VendorName .
                    ?VendorProductDeatils eeo:hasVendorProductPrice ?price ;
                            eeo:hasProductProductQuantity ?stock .
        """
        query += f"filter contains(lcase(?title),'{keyword.lower()}')"
        query += "\n}"
        sparql = SPARQLWrapper(
            "https://product-sparql001.herokuapp.com/ds/sparql"
        )
        sparql.setReturnFormat(JSON)
        sparql.setQuery(query)
        results = sparql.query().convert()

        try:
            ret = sparql.queryAndConvert()
            Title = []
            ModelNumber = []
            Image = []
            for r in ret["results"]["bindings"]:
                Title.append(r["title"]['value'])
                ModelNumber.append(r["modelNumber"]['value'])
                Image.append(r["image"]['value'])
        except Exception as e:
            print(e)

        Title = np.unique(Title)
        Image = np.unique(Image)
        ModelNumber = np.unique(ModelNumber)
        ModelNumber = [urllib.parse.quote(x, safe='') for x in ModelNumber]
        details = zip(Title, ModelNumber, Image)
        return render(request, 'product.html', context={"detail": details})
    return render(request, 'index.html')
```

Fig. 11 Keyword product search

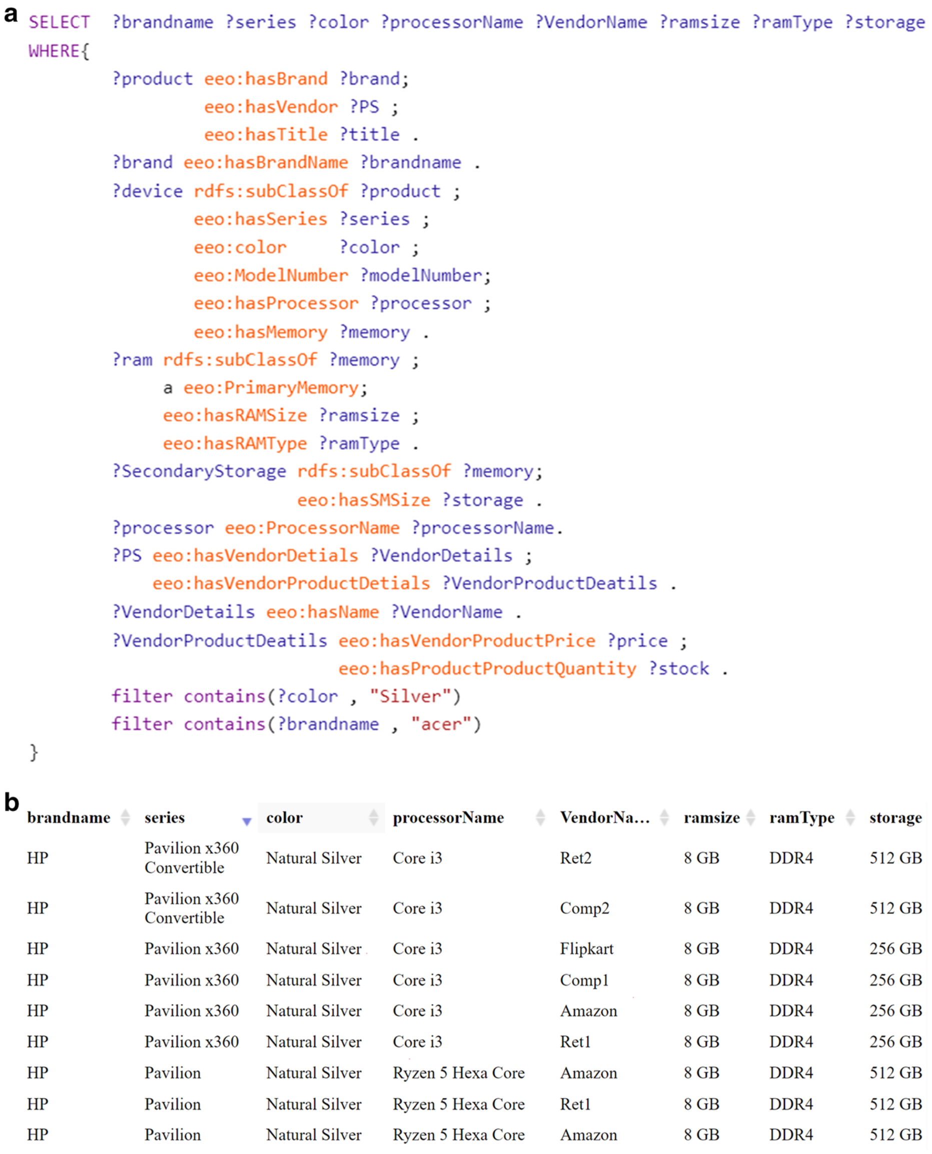


https://product-sparql001.herokuapp.com/ds/sparql

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT * WHERE {
  ?sub ?pred ?obj .
} LIMIT 10
```

Fig. 12 SPARQL endpoint

- **Data Integration:** Many web product search portals provide parametric search with basic product properties like brand, reviews, and price and that too only for one vendor (Amazon, Flipkart, Google shopping). As soon as some feasible matching is found, the user moves to find the least cost product for that specification. [46] proposed an ontology-based product search and exploration solution but did not use standardized vocabularies for annotating product information. In contrast, our aim is to provide an aggregator semantic web portal for Web-wide search and exploration to provide aggregated product information in one unified view. Users can ask even the queries that were impossible over the individual datasets, like, "Provide a list of all Notebooks with RAM 8 GB, price less than equal to 800 Euros and available in stock".
- **Interoperability:** Our work is better than any existing aggregator sites also because such sites process RDFa annotated (X)HTML pages coming from eShops and aggregate product information [42]; while we provide service endpoint to vendor also for uploading their product catalogues in any heterogeneous format. Moreover, existing aggregator sites like Shopping.com and Shopzilla have their own application programming interfaces (APIs) with their own standards; it gets difficult and costly for eShops and retailers to customize their data according to their standards. Our aggregator site represents and stores data in a universal abstract semantic data model (RDF) annotated through our developed ontology EEO, which in turn utilizes the accepted standard vocabularies. This helps in consensus understanding – sharing common understanding of the domain knowledge among people and among processing agents by making the data independent of its internal representations.
- The major goal of existing product matching and categorization works is finding out whether two offers on different Web pages are referring to the same product; while our goal is to provide search facility on an aggregator portal where products from all the vendors are already available [[33], Part et al. 2006].
- The long-term sustainability and viability of businesses is not clear, because various fees are charged to vendors by the eShops. Therefore, this proves to be ***better than the alternative solutions.*** We simplify the application development by moving most of the application logic and analytics into the triple store. We structure the information on the Web right at the time of authoring. This later facilitates the automated deployment. The cloud-ready services and the REST APIs enable analysis on big datasets also with added performance. The service endpoints s***upport more reliable and less ambiguous communication.*** Disparate companies and different groups of people use different nomenclature



a

```
SELECT  ?brandname ?series ?color ?processorName ?VendorName ?ramsize ?ramType ?storage
WHERE{
        ?product eeo:hasBrand ?brand;
                 eeo:hasVendor ?PS ;
                 eeo:hasTitle ?title .
        ?brand eeo:hasBrandName ?brandname .
        ?device rdfs:subClassOf ?product ;
                eeo:hasSeries ?series ;
                eeo:color     ?color ;
                eeo:ModelNumber ?modelNumber;
                eeo:hasProcessor ?processor ;
                eeo:hasMemory ?memory .
        ?ram rdfs:subClassOf ?memory ;
             a eeo:PrimaryMemory;
             eeo:hasRAMSize ?ramsize ;
             eeo:hasRAMType ?ramType .
        ?SecondaryStorage rdfs:subClassOf ?memory;
                          eeo:hasSMSize ?storage .
        ?processor eeo:ProcessorName ?processorName.
        ?PS eeo:hasVendorDetials ?VendorDetails ;
            eeo:hasVendorProductDetials ?VendorProductDeatils .
        ?VendorDetails eeo:hasName ?VendorName .
        ?VendorProductDeatils eeo:hasVendorProductPrice ?price ;
                              eeo:hasProductProductQuantity ?stock .
        filter contains(?color , "Silver")
        filter contains(?brandname , "acer")
}
```

b

brandname	series	color	processorName	VendorNa…	ramsize	ramType	storage
HP	Pavilion x360 Convertible	Natural Silver	Core i3	Ret2	8 GB	DDR4	512 GB
HP	Pavilion x360 Convertible	Natural Silver	Core i3	Comp2	8 GB	DDR4	512 GB
HP	Pavilion x360	Natural Silver	Core i3	Flipkart	8 GB	DDR4	256 GB
HP	Pavilion x360	Natural Silver	Core i3	Comp1	8 GB	DDR4	256 GB
HP	Pavilion x360	Natural Silver	Core i3	Amazon	8 GB	DDR4	256 GB
HP	Pavilion x360	Natural Silver	Core i3	Ret1	8 GB	DDR4	256 GB
HP	Pavilion	Natural Silver	Ryzen 5 Hexa Core	Amazon	8 GB	DDR4	512 GB
HP	Pavilion	Natural Silver	Ryzen 5 Hexa Core	Ret1	8 GB	DDR4	512 GB
HP	Pavilion	Natural Silver	Ryzen 5 Hexa Core	Amazon	8 GB	DDR4	512 GB

Fig. 13 **a** SPARQL Query for EER (with Input parameters “acer”, Silver color laptop) **b** Output generated by EER (with Input parameters “acer”, Silver color laptop)

for the same thing. By establishing a common technical language, eShops as well as end users will be able to process and analyze data automatically.

- ***A better search and discovery of products/services is provided across the vendors.*** The smart recommender recommends a list of equipment available based on the input specifications. For every product or service required, a lot of products and services are available in the market and that too provided by a number of vendors. It becomes difficult for a customer to select which product/service to purchase and from which vendor. It helps the manufacturer also to select equipment to market at specified avenues. If the requirement of the user (/avenue) is known in the form of specifications, the system can generate recommendations on which equipment best fits the requirement specification.

- ***We harness the semantics associated with the multimedia attributes of products like the images and the product promotional videos.***
- **Reusability and Richer Queries:** The web ontology for products (EEO) has been developed as a compatible extension of three controlled vocabularies, namely schema.org, GoodRelations ontology, and the Dublin Core. This provides for the necessary ***interoperability and reusability***. Categorizing the things globally and storing them at a common place promise reuse of specifications and avoids duplication of work; ***leading to cost reductions.*** Through knowledge graphs, the related data is now linked, hence ***providing relational synergy***. The user comes to know additional related information that was not possible in the traditional approach, through richer queries with data brought from other linked data sources using simple reasoning.

6 Conclusion and directions for future work

The product descriptions on semantic web are inherently complex and in heterogeneous formats. This work has provided a common comprehensive platform for representing and storing different product descriptions. We have exploited the knowledge graphs for understanding far-flung attributes of products to calculate similarity with the input specifications. The graph representation is abstract and independent of the details of the native structures, therefore is schema independent; a change in the local database schemas, HTML structures, etc. does not affect the knowledge graph.

There are certainly some aspects and properties that have not yet sufficiently been addressed by the proposed framework. However, we present here an outlook into the future that we envision.

- The said EEO has been developed with the aim of facilitating product discovery. Meanwhile, we can see that we have provided descriptive information of every product instance including the capabilities, technical features, the consumers' understanding of such devices, and ability to operate them correctly and effectively. One service endpoint can be provided to understand the features and capabilities of any product and visualize the detailed product specifications and description. The existence of the knowledge graph will facilitate discovering semantic relationships among various product instances.
- Let us give a more prospective outlook for the evaluation. We anticipate use cases scenarios when actual real-world data is available and the developed application is deployed on the web. This will be the time when it will be evaluated against the benchmarks.
- We require providing a rationale for the discovery of products for building trust. This is done through explainable recommendations [11, 45]. The approaches to explainable AI have problems. This can be done using OWL ontology and writing rules for explanations. We will extend in this direction.

- The current version is more of a product discovery than recommendation. By having a personalized knowledge graph of the user's behavior history, personalized product recommendations could be provided.
- As it is the in-progress project, the multimedia concept is currently being implemented using which the user can search for the related product by uploading the front and back image of the product. This feature will help the users who have very vague ideas regarding the product and lack product specification knowledge. As the foundation is now in place, incorporating this feature should not be problematic.
- We plan to allow the vendor to input product catalogue images with brief natural language descriptions of their products. The image descriptions will be represented using RDF and stored in our triple store using NLP tools. When the image descriptions are represented as RDF triples, the associated semantics makes it easier to classify and search products.

Acknowledgments The author acknowledges the support of her three scholars Mr., Vishal Lama, Ms. Pooja Harde, and Mr. Ankush Bisht for critical discussions during the writing of the manuscript.

Data availability The code and datasets generated during and/or analysed during the current study are available on the GitHub Repository with link: https://github.com/semintelligence/Product-Discovery.

Declarations

Conflict of interests The author declares no conflict of interest (financial or non-financial).

References

1. Akritidis L, Bozanis P (2018) Effective unsupervised matching of product titles with k-combinations and permutations. In: 2018 Innovations in Intelligent Systems and Applications (INISTA). IEEE, pp 1–10
2. Allan J, Carbonell JG, Doddington G, Yamron J, Yang Y (1998) Topic detection and tracking pilot study final report
3. Bhatt S, Sheth A, Shalin V, Zhao J (2020) Knowledge graph semantic enhancement of input data for improving AI. IEEE Internet Comput 24(2):66–72
4. Bhutani P, Baranwal SK, Jain S (2021) Semantic framework for facilitating product discovery. In: 2021 Advances in Computational Intelligence, its Concepts & Applications (ACI 2021), vol. 2823. CEUR-WS, pp 30–36
5. Bolelli L, Ertekin Ş, Lee Giles C (2009) Topic and trend detection in text collections using latent dirichlet allocation. In: European Conference on Information Retrieval. Springer, Berlin, Heidelberg. pp. 776–780
6. Brunner JS, Ma L, Wang C, Zhang L, Wolfson DC, Pan Y, Srinivas K (2007) Explorations in the use of semantic web technologies for product information management. In: Proceedings of the 16th international conference on World Wide Web, pp 747-756
7. Calvanese D, Cogrel B, Komla-Ebri S, Kontchakov R, Lanti D, Rezk M, Rodriguez-Muro M, Xiao G (2017) Ontop: answering SPARQL queries over relational databases. Semant Web 8(3):471–487
8. Chaves-Fraga D, Priyatna F, Santana-Pérez I, Corcho O (2018) Virtual statistics knowledge graph generation from CSV files. In ISWC (best workshop papers). pp. 235-244
9. Curry E, Ojo A (2020) Enabling Knowledge Flows in an Intelligent Systems Data Ecosystem. In: Real-time Linked Dataspaces. Springer, Cham, pp 15–43
10. Dalle Lucca Tosi M, dos Reis JC (2020) Understanding the evolution of a scientific field by clustering and visualizing knowledge graphs. J Inf Sci 48:71–89. https://doi.org/10.1177/0165551520937915
11. Diaz-Agudo B, Caro-Martinez M, Recio-Garcia JA, Jorro-Aragoneses J, Jimenez-Diaz G (2019, September) Explanation of recommenders using formal concept analysis. In: International conference on case-based reasoning. Springer, Cham. pp. 33-48

12. Dubey S, Patel A, Jain S (2021) Conversion between semantic data models: the story so far, and the road ahead. In: Jain et al (eds) Web Semantics – Cutting Edge and Future Directions in Health Care. Elsevier Publishing
13. Duvvuru A, Radhakrishnan S, More D, Kamarthi S, Sultornsanee S (2013) Analyzing structural & temporal characteristics of keyword system in academic research articles. Procedia Comput Sci 20:439–445
14. Fitzpatrick D, Coallier F, Ratté S (2012) A holistic approach for the architecture and design of an ontology-based data integration capability in product master data management. In: IFIP International Conference on Product Lifecycle Management. Springer, Berlin, Heidelberg. pp. 559–568
15. García-González H, Boneva I, Staworko S, Labra-Gayo JE, Lovelle JMC (2020) ShExML: improving the usability of heterogeneous data mapping languages for first-time users. PeerJ Comput Sci 6:e318
16. Griffiths TL, Steyvers M (2004) Finding scientific topics. Proc Natl Acad Sci 101(suppl 1):5228–5235
17. Hepp M (2005) eClassOWL: A fully-fledged products and services ontology in OWL. Poster Proceedings of ISWC2005. Galway
18. Hepp M (2008) Goodrelations: An ontology for describing products and services offers on the web. In: International conference on knowledge engineering and knowledge management. Springer, Berlin, Heidelberg. pp. 329–346
19. Iglesias E, Jozashoori S, Chaves-Fraga D, Collarana D, Vidal ME (2020, October). SDM-RDFizer: an RML interpreter for the efficient creation of rdf knowledge graphs. In: Proceedings of the 29th ACM international conference on Information & Knowledge Management. pp. 3039-3046
20. Jain S, Grover A, Thakur PS, Choudhary SK (2015) Trends, problems and solutions of recommender system. In: International Conference on Computing, Communication & Automation. IEEE, pp 955–958
21. Jain S, Jain V, Balas VE (Eds) (2021) Web semantics – cutting edge and future directions in health care. Elsevier Publishing
22. Jozashoori S, Chaves-Fraga D, Iglesias E, Vidal ME, Corcho O (2020) Funmap: efficient execution of functional mappings for knowledge graph creation. In: International Semantic Web Conference. Springer, Cham, pp 276–293
23. Junior AC, Debruyne C, Brennan R, O'Sullivan D (2016, November). FunUL: a method to incorporate functions into uplift mapping languages. In: Proceedings of the 18th international conference on information integration and web-based applications and services. pp. 267-275
24. Kim H (2021) Developing a product knowledge graph of consumer electronics to manage sustainable product information. Sustainability 13(4):1722
25. Li X, Chen C-H, Zheng P, Wang Z, Jiang Z, Jiang Z (2020) A knowledge graph-aided concept–knowledge approach for evolutionary smart product–service system development. J Mech Des 142(10)
26. Li X, Lyu M, Wang Z, Chen CH, Zheng P (2021) Exploiting knowledge graphs in industrial products and services: a survey of key aspects, challenges, and future perspectives. Comput Ind 129:103449
27. Lin Y-C, Das P, Trotman A, Kallumadi S (2019) A dataset and baselines for e-commerce product categorization. In: Proceedings of the 2019 ACM SIGIR international conference on theory of information retrieval (ICTIR '19). Association for Computing Machinery, New York, NY, USA, 213–216. https://doi.org/10.1145/3341981.3344237
28. Mauge K, Rohanimanesh K, Ruvini J-D (2012) Structuring e-commerce inventory. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1. Association for Computational Linguistics. pp. 805–814
29. Meusel R, Petrovski P, Bizer C (2014) The webdatacommons microdata, rdfa and microformat dataset series. In: International Semantic Web Conference. Springer, Cham. pp. 277–292
30. Meusel R, Primpeli A, Meilicke C, Paulheim H, Bizer C (2015) Exploiting microdata annotations to consistently categorize product offers at web scale. In: International Conference on Electronic Commerce and Web Technologies. Springer, Cham. pp. 83–99
31. Nederstigt LJ, Aanen SS, Vandic D, Frasincar F (2014) FLOPPIES: a framework for large-scale ontology population of product information from tabular data in e-commerce stores. Decis Support Syst 59:296–311
32. Nigam P, Song Y, Mohan V, Lakshman V, Ding WA, Shingavi A, Teo CH, Gu H, Yin B (2019) Semantic product search. In: Proceedings of the 25th ACM SIGKDD international conference on Knowledge Discovery & Data Mining (KDD '19). Association for Computing Machinery, New York, NY, USA, 2876–2885. https://doi.org/10.1145/3292500.3330759
33. Noy NF, Musen MA (2003) The PROMPT suite: interactive tools for ontology merging and mapping. Int J Hum Comput Stud 59(6):983–1024
34. Osborne F, Motta E (2012) Mining semantic relations between research areas. In: International Semantic Web Conference, Springer, Berlin, Heidelberg, pp. 410–426
35. Patel A, Jain S (2021) Present and future of semantic web technologies: a research statement. Int J Comput Appl 43(5):413–422

36. Pohorec S, Zorman M, Kokol P (2013) Analysis of approaches to structured data on the web. Comput Stand Interfaces 36(1):256–262
37. Roos T, Myllymaki P, Tirri H (2002) A statistical modeling approach to location estimation. IEEE Trans Mob Comput 1(1):59–69
38. Sehgal S, Chaudhry S, Biswas P, Jain S (2016) A new genre of recommender systems based on modern paradigms of data filtering. Procedia Comput Sci 92:562–567
39. Sheth A, Fisher M (2004) Semantic Enterprise Content Management. https://doi.org/10.1201/9780203507223.ch9
40. Sjekavica T, Obradović I, Gledec G (2013) Ontologies for multimedia annotation: an overview. In: 4th European conference of computer Science (ECCS'13)
41. Stolz A, Rodriguez-Castro B, Hepp M (2013) Using BMEcat catalogs as a lever for product master data on the semantic web. In: Extended Semantic Web Conference. Springer, Berlin, Heidelberg, pp. 623–638
42. Vandic D, Van Dam JW, Frasincar F (2012) Faceted product search powered by the semantic web. Decis Support Syst 53(3):425–437
43. Wauer M, Schuster D, Meinecke J (2010) Aletheia: an architecture for semantic federation of product information from structured and unstructured sources. In: Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services, pp. 325–332
44. Wu J, Choudhury SR, Chiatti A, Liang C, Giles CL (2017) HESDK: A hybrid approach to extracting scientific domain knowledge entities. In: 2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL), IEEE, pp. 1–4
45. Xian Y, Fu Z, Muthukrishnan S, De Melo G, Zhang Y (2019, July) Reinforcement knowledge graph reasoning for explainable recommendation. In: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval. pp. 285-294
46. Zhang L, Zhu M, Huang W (2009) A framework for an ontology-based e-commerce product information retrieval system. J Comput 4(6):436–443
47. Zhang Y, Lu J, Liu F, Liu Q, Porter A, Chen H, Zhang G (2018) Does deep learning help topic extraction? A kernel k-means clustering method with word embedding. J Informetr 12(4):1099–1117

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.