S.A.R.A (Smart AI Refrigerator Assistant)

Sachin Singh Bhadoria[1], Saniya Kirkire[1], Rut Vyas[1], Satvik Deshmukh[1], Yukti Bandi[2]
[1] U. G. Student, Department of Electronics and Telecommunication, D.J Sanghvi College of Engineering, Vile Parle (W), Mumbai- 400056
[2] Assistant Professor, Department of Electronics and Telecommunication, D.J Sanghvi College of Engineering, Vile Parle (W), Mumbai- 400056
E-mail: [1]sachinsinghbhadoria1@gmail.com, [1]saniyakirkire@gmail.com, [1]rutvyas@gmail.com, [1]satvikdesmukh3@gmail.com, [2]yukti.bandi@djsce.ac.in

***Abstract*—The advent of new technologies has dramatically revolutionized our lifestyles. The traditional approach towards the development of devices like cell phones, televisions and watches has now been enhanced, improvised, and modernized with novel technologies like AIML, IOT to realize the modern-day ideas of smart-phones, smart-TVs, and smart-watches. On a similar basis, the aim of this project is to develop an AI powered assistant which can be integrated with conventional fridge like a plug and play device to introduce some innovative features to the modern-day refrigerators. This paper presents the main feature of this assistant, recipe recommendation, a functionality that has been achieved through extensive Natural language processing.**

Keywords— AIML, Natural Language Processing, Data mining, Web Scraping, Recipe Recommendation, Tokenization, Web Development

I. INTRODUCTION

The fast pace of life has increased the popularity of instant foods. Further, the recent quarantine period has piqued the interests of many in cooking. When it comes to food, deciding what recipe to make often takes longer than the actual preparation time. To save this time and present a wide variety of options to the user, S.A.R.A, an AI-powered refrigerator assistant has been developed. Its features encompass Recipe recommendation based on the user-input. This assistant can interact with the user through a progressive web application while the data processing was done remotely in a server. The data required for the recommendation algorithm was acquired by leveraging the web scraping technologies using Open-source python libraries. This data was then stored in Json files. The textual data was organized with Data wrangling and analysis techniques. The recipe recommendation functionality was achieved by developing a search engine specific to food recipes through extensive Natural language processing. Further, the technologies used were completely Open-source and the modifications which will be required to integrate our assistant are minimal as a result of which the whole idea is extremely budget friendly. Introducing such a feature will not just save time but also entice the user to try newer recipes. The user can explore and include these newly discovered recipes in his or her diet. The goal is not only restricted to comfort and ease but also aims towards the accomplishment of a healthier lifestyle. In addition to this, a variety of additional features can be introduced in this assistant which will ensure even more ease and benefit for the user. This paper provides the detailed approach, methodology, system architecture and an in-depth analysis of the functionality this assistant is capable of. Results that were achieved with this solution are also presented here.

II. LITERATURE SURVEY

Since the early 2000s, one has been fascinated by the idea of connecting all of one's daily life utilities to the internet (IoT). Day by day, people want to spend less time on menial jobs and want everything connected to their smartphones. This led to LG launching the world's first smart refrigerator called the Internet Digital DIOS in June 2000 [1]. Although this was an unsuccessful venture due to its high costs and unnecessary features, this instigated other companies to enter this field and develop their own technologies. One of the technologies implemented was a recipe recommendation system based on ingredient availability and survey of such work by some of the brilliant pioneers has been carried out.

Zheng Xian Li and team proposed a personalized hybrid recommendation system that is an amalgamation of model-based CF algorithm and content-based filtering as a supplement to increase the accuracy of recipes recommended. They made use of Apache Spark as the engine for recommendation, MySQL for storing and managing recommendation results or metadata and Hadoop Distributed File System (HDFS) to manage unstructured data [2]. Their experiments displayed that the recipe suggestion system has scalable computational capability to process massive information of recipes.

Diya Lu and team proposed a Recipe Search based on nutritional value. Their primary focus was to display pre-trained recipe embeddings that yielded more diverse results in comparison to searches based on keyword. They adopted

the cosine similarity to measure the distance between two recipe embeddings and sacrificed accuracy for speedy retrieval by deploying top-k approximate nearest neighbours (ANN) [3]. However, their approach of searching first then choosing with additional nutrition information is a naive way to do nutrition guided recipe search and further research can be made in this domain.

Zhen Feng Lei and team focused on recipe suggestions accompanied by logical interpretations produced from images or videos. They proposed a multi-modal recipe recommendation system via the knowledge graph (RcpMKR). Further, they constructed a recipe knowledge graph (RcpKG) using multi-modality and hierarchical thought and inculcated BERT-based multi-modal models to generate explanations [4]. Thus, the proposed extensible multi-modal recipe fusion framework provided guidelines for improving performance of recipe recommendation and generating reasonable and acceptable explanations derived from images or videos related to the recommended results.

Vasvi Bajaj and team made use of the Graph Database of the NoSQL family of databases to recommend recipes based on the ingredients selected in the query [5]. They loaded the Python crawled data into a Neo4j database through R-programming script. The nodes were made up of recipe and ingredient names while the edges represented the relationships between the nodes. They concluded that graph databases are significantly faster as they reduce the time required to compute "JOIN" operation related queries which need to be executed in other NoSQL's and Relational Databases like RDBMS. However, such a system does not consider the weight of the ingredients with respect to the recipe and therefore is a very crude and not so customer friendly recipe recommendation system.

III. IMPLEMENTATION

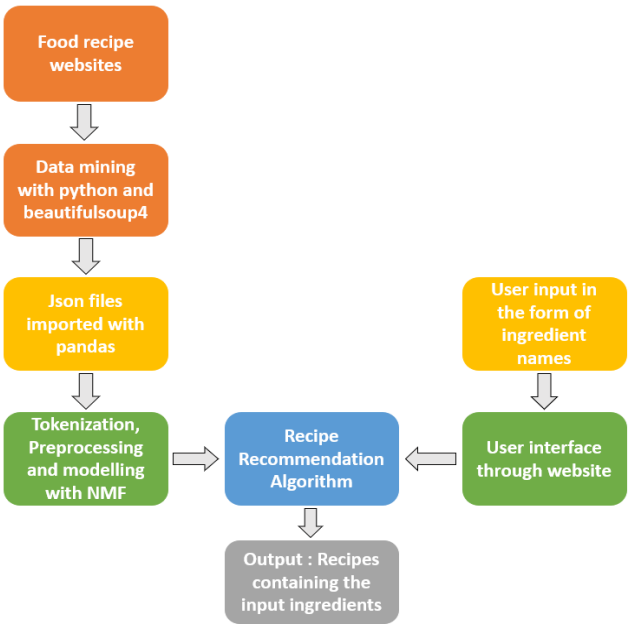

Fig.1-Implementation flowchart

The main pipeline of the system works on a singular directional system flow; the algorithm getting input from the users in the form of an array of the ingredients. The algorithm appends the top 3 recipes which can be made using the ingredients input by the user. The matching and ranking of recipes with respect to ingredients was achieved through extensive natural language processing during the training process for the algorithm. The data required to train the model and to build this search engine was gathered from three different food blogging websites with data mining techniques and web scraping technologies. To integrate an API with the algorithm and to ensure that the output is sent back to the client, a web server that could be integrated easily with our system was required.

The working process of the whole project is divided into the following steps.

A. Data mining

Collection of data was a crucial element in building the algorithm which is the backbone of our assistant. Adequate amount of organised data was needed to make sure that the algorithm is capable of generating useful patterns and ensure an optimum state where the model is neither overfitting nor underfitting. Manually generating a huge quantity of data consumes a lot of time, human effort and computational resources. To tackle this problem, "BeautifulSoup", a python-based web scraping library was used in the first stage. It is capable of automating the entire process of data extraction from the internet website efficiently out of XML and HTML files. BeautifulSoup works on a parse tree methodology that is very similar to the tree data structure. It allows the code to browse multiple pages concurrently in a hierarchical format owing to its structure.

The data from three different culinary websites was extracted using beautifulsoup4, spacy and other open-sourced Python libraries and tools. The URL of these websites was fed to the scrapper which was used to extract the data including the title of the recipe, its ingredients, a picture, and the cooking instructions from the HTML web pages. The websites used were:

a) Food Network
b) All recipes
c) Epicurious

Multithreading was utilized to make sure this was done effectively and resulted in a total data collection of over 100 thousand recipes. The data was stored in JSON file format and later imported with pandas for further processing and analysis. However, the scraping resulted into multiple data errors and required heavy pre-processing in order to be used for further algorithm implementation.

B. Pre-processing and Tokenization

The JSON files consisting the raw data were directly obtained from websites through web scraping due to which many anomalies were introduced. In order to use the data for training, a considerable amount of cleaning was necessary. At this stage of implementation in the Jupyter notebook, the raw data stored in json files was imported using open-source libraries like pandas and NumPy. First, all the tuples with missing fields and the ones that contained links for recipe pictures were removed. Then, a lower limit of 20 characters was set for the instruction attribute and recipes that failed to meet this requirement were dropped on grounds of insufficient information. While extracting data, the term 'ADVERTISEMENT' was included in each ingredient attribute due to ad section of webpages. This term had to be dropped. Finally, several unnecessary elements of the data entities including words without any information value, punctuations, digits, symbols, and unwanted tabs/new lines as well as irregular entries in the data frame with absence of some or the other major entities were removed. As the entire algorithm works on matching documents with the ingredient or recipe name fed by the user, the most suitable method was to combine all three columns of title, ingredients, and instruction into singular textual components.

Extracting information from text is difficult and hence the system needs to break the sentences into smaller chunks which are capable of being processed. There are various NLP techniques to obtain tokenized words from sentences. In the tokenization stage for our project, spacy tokenizer was incorporated as it was comparatively faster and provided an ease of customization. Python's multiprocessing library was used for this heavy process. Here, the entire raw text was broken down into words that could be analysed and correlated to each other for the purpose of assigning weights. The textual data was lemmatized and stop words were removed from the data. This tokenized text was further vectorized with tfidf vectorizer to assign weights as per the relative significance of words in the document.

C. Creation of Word Embeddings

Conversion of these blocks of texts into quantitative figures is essential to draw accurate predictions. Word embeddings are used for this purpose. These are a type of word representations that allows words with similar meaning to have a similar representation. In case of recipes, it is required to find the relevance of each word inside a corpus which will fetch the user recipes relevant to the ingredients they selected. Here, "Term Frequency- Inverse Document" Frequency (TFIDF) was used for vectorization of words. This is done to find word frequencies of important words, and discard words that appear a lot but have negligible meaning: Example: "The", "a", "an", etc. The meaning increases with the number of times a word appears in a text data series, but it is compensated by its word frequency in the dataset. This was done using the TfidfVectorizer function provided by sklearn themselves. It provided generic weights to all the topics, which will relate to the frequency of these topics appearing inside the entire document set. In simpler words, TFIDF is a simple frequency score that tries to highlight words that are more interesting and important. The tokenized text was passed to fit inside this vectorizer and was later used to extract feature names, or the highlighted words. The TFID formula is as follows,

$$W_{x,y} = tf_{x,y} \times \log(\frac{N}{df_x}) \quad (1)$$

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = Total number of documents

D. Topic Modelling

Topic Modelling is defined as "type of statistical modelling for discovering the abstract "topics" that occur in a collection of documents". It is an unsupervised approach used for finding and observing a bunch of words (called "topics") in large clusters of texts. Topics can be defined as "a repeating pattern of co-occurring terms in a corpus". A good topic model should result in – "health", "doctor", "patient", "hospital" for a topic – Healthcare, and "farm", "crops", "wheat" for a topic – "Farming". In the present case, these topics were the recipes and the words would be the ingredients. There are two algorithms that can be used for the topic modelling process which are well received by the community namely LDA [7] (Latent Dirichlet Allocation) and NNMF [6] (Non-Negative Matrix Factorization). The topic modelling algorithm which was chosen for this application was NNMF. The choice was made based on the document score and stability of convergence. The document score refers to the number of accurate predictions one topic can do for x number of documents. The graphs for three of the topics were plotted for both LDA and NNMF.

NNMF decomposes (or factorizes) high-dimensional vectors into a lower-dimensional representation. These lower-dimensional vectors are non-negative which also means their coefficients are non-negative.

Using the original matrix (A), NMF will give you two matrices (W and H). W is the topics it found, and H is the coefficients (weights) for those topics. In other words, A is articles by words (original), H is articles by topics and W is topics by words. The matrix formed was later passed for Text Ranking.

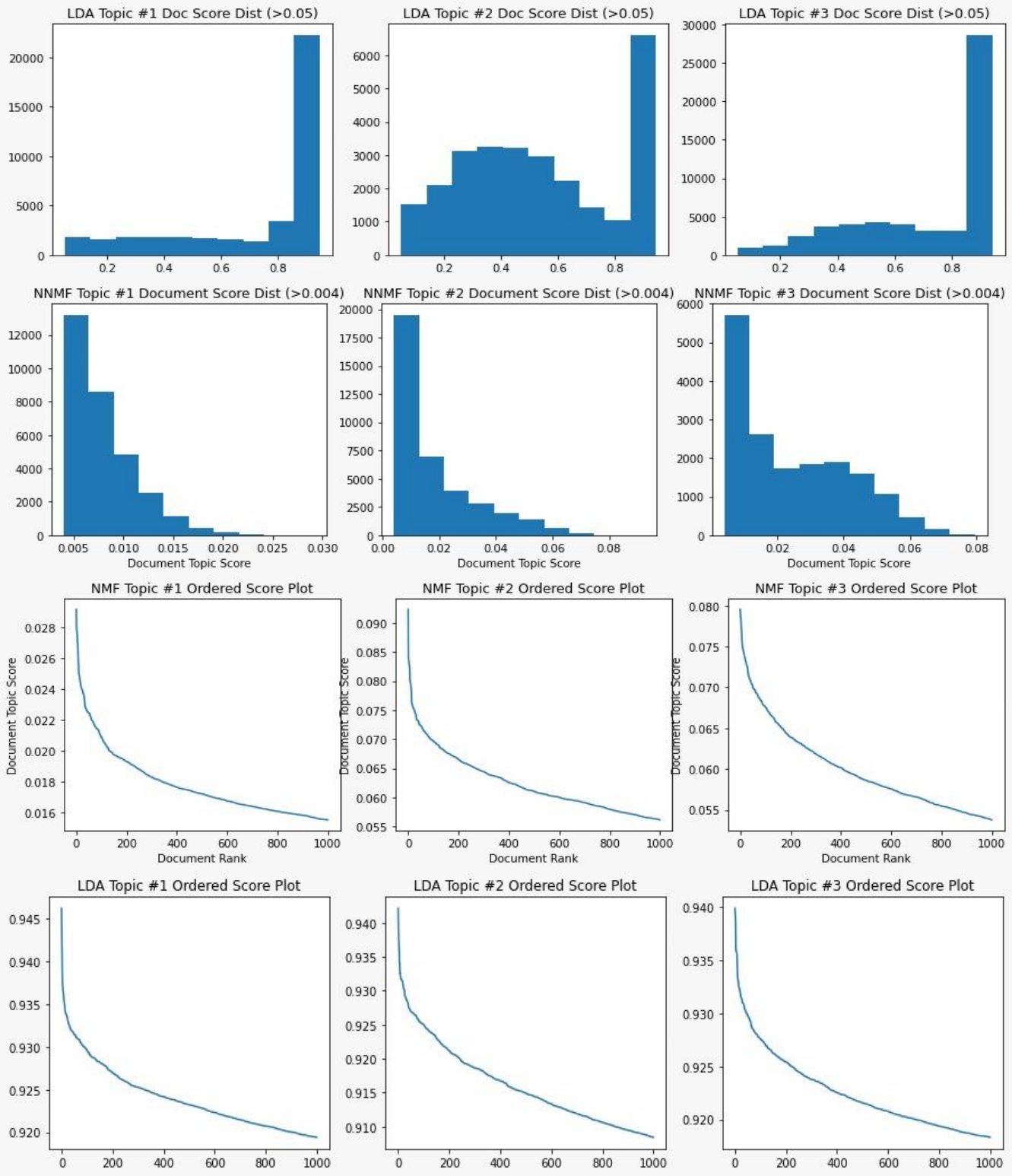

Fig.2 Plot of LDA vs NNMF for accuracy of algorithm with topic matching

It was observed that as the document score increased, the number of documents that could be covered decreased. On comparing the above graphs, it could be concluded that

1. NNMF required lesser computational power on the workbench when training and provided the ability to train for higher epochs.
2. NNMF was used as it provided a good convergence and had more distinction on topics.
3. NNMF showcased a levelling out on 200 documents, therefore as per the objective function 200 was chosen as the document cut off.

E. Text Ranking

Text Ranking exists of two functionalities in general: Topic Extraction and Keywords Extraction. The top-hundred documents ranked in similarity among topics were extracted, as after this count the similarities between them starts reducing. These were sorted in a descending order to rank the higher similarities above. As recipes are not traditional texts, they have to contain manually entered 'stop words. Thus, a couple of custom stop words for this specific use case were made. This set of stop words can be updated as per requirement of the algorithm.

As adjacency table was extracted from the recipes, the window was limited to 4 words at most, allowing the algorithm to be precise and not work with long strings. This matrix was later passed into a text ranking library called network. This generated a well-documented cluster, where one was clearly able to find patterns between the keywords belonging in topics. Score distribution analysis was done for NNMF algorithm, and the results showcased that the range of keywords covered under the text topic 'baking' extended to up to 10,000 of the top ranked recipes falling in that category.

F. Querying Algorithm

A compilation of a system that can be used to extract recipes matching the keywords was referred as the 'querying algorithm'. This resulted in forming a compiled version of the above procedure into a storable file using Panda's library. Tags were generated for all the recipes as per their rankings with the keywords, these tags would later be utilized by the queries to find recipes. These keywords were indexed to shorten the searching time, much like a no-SQL database.

The query algorithm took an array of ingredients as input, these inputs were then split into different documents, these documents went through a vectorizer sharing the same properties of that in the modelling procedure. The vectorized ingredients, or keywords, were sent into a recursive function to break them down into NumPy arrays in descending weights. This NumPy array was multiplied with the previously obtained vectorized ingredients to obtain index form, which was added together using a reducer. This final vector was then used to search inside the tokenized and vectorized database to find the most appropriate score from the list and return that as the recipe to cook.

IV. RESULTS

This query algorithm was hosted on a local system using a flask server to utilize fast development speeds. This can easily be transferred in an AWS lambda function during the production stage. The front end was currently made with a JavaScript framework called 'Next.js', which takes in three ingredients and sends it to the flask backend using REST API. The response sent back contains an array of JSON objects. These objects are the top-3 recipes recommended by the system and will be rendered into the UI seamlessly. A picture to be displayed with the recipe was also reverse searched using the Google Custom Search API.

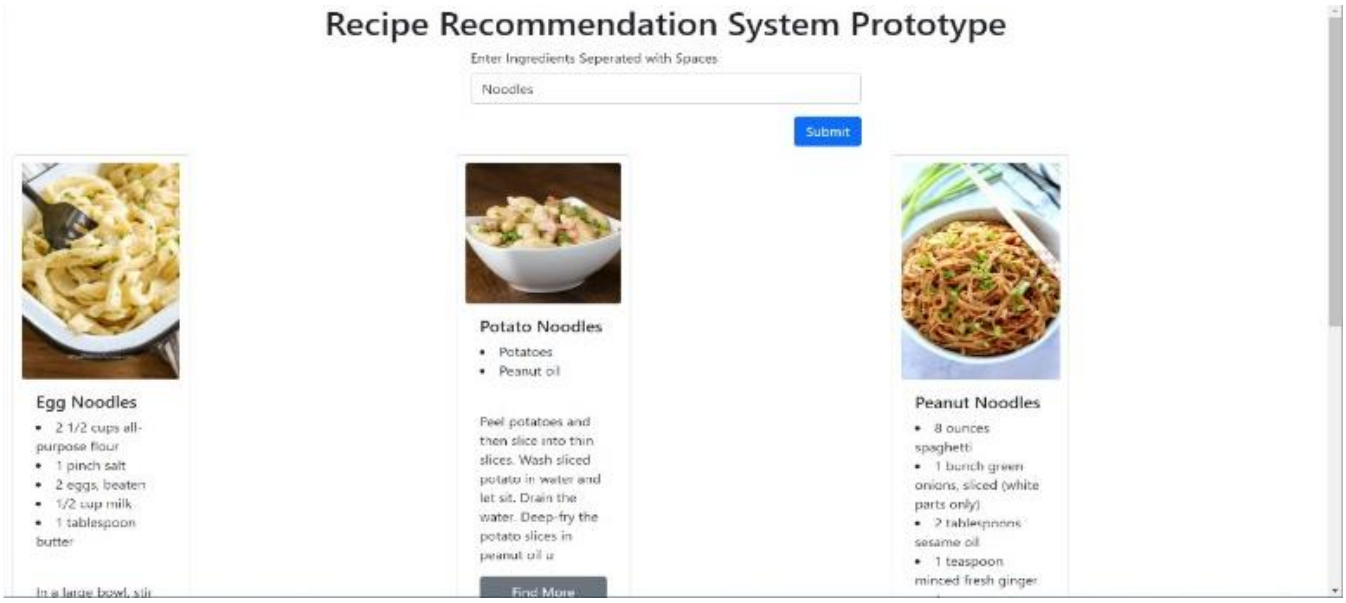

Fig.3 - query algorithm hosted on a local system using a flask serve

V. CONCLUSION

The Recipe Recommendation system was developed using data gathered by leveraging web scraping technologies. The tools used during the cleaning, pre-processing, tokenization and modelling were completely Open Source. The Query algorithm is similar to a search engine. It takes an array of ingredients as its input and returns the top 3 recipes which can be made using these input ingredients. A front-end web application was designed and implemented to demonstrate the actual deployment of the assistant.

VI. FUTURE SCOPE

Apart from Recipe recommendation, several additional features can be integrated with our assistant. The assistant can interact with its user through a mobile application. One of the features which could be added is decay detection. Currently, the user needs to manually enter the the items stored in the refrigerator. Using OpenCV and by including a camera as an additional hardware device, object detection can be used for automatically tracking all the items being stored. Further by tracking the date and time of storage, it will be possible to monitor if an item is being stored for unusually longer periods and alert the user in case this is observed. For instance, if curd is stored for more than a day, then the assistant will alert the user. This will prevent food items from going bad. Alternatively, RFID tags could also be used. Tracking items can also be used to alert the user in case stock of a particular frequently consumed item is about to get over. It can also provide a way to monitor the overall diet of the user depending what type of food items are frequently consumed. This may help the user in interpreting the overall nutritional value his or her diet provides. In addition to this, using block chains, it may also be possible to automatically order the items which are frequently stocked by the user. For this the assistant will need to track the frequency with which order for a particular item is placed by the user.

VII. REFERENCES

[1] ^ *"LG UNVEILS INTERNET-READY REFRIGERATOR"*. LG Electronics. telecompaper.com. June 21, 2000.

[2] Z. Li, J. Hu, J. Shen and Y. Xu, "A Scalable Recipe Recommendation System for Mobile Application," 2016 3rd International Conference on Information Science and Control Engineering (ICISCE), 2016, pp. 91-94, doi: 10.1109/ICISCE.2016.30.

[3] D. Li, M. J. Zaki and C. -H. Chen, "Nutrition Guided Recipe Search via Pre-trained Recipe Embeddings," 2021 IEEE 37th International Conference on Data Engineering Workshops (ICDEW), 2021, pp. 20-23, doi: 10.1109/ICDEW53142.2021.00011.

[4] Zhenfeng Lei, Anwar Ul Haq, Adnan Zeb, Md Suzauddola, Defu Zhang, Is the suggested food your desired: Multi-modal recipe recommendation with demand-based knowledge graph, Expert Systems with Applications, Volume 186, 2021,115708, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2021.115708.

[5] V. Bajaj, R. B. Panda, C. Dabs and P. Kaur, "Graph Database for Recipe Recommendations," 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2018, pp. 1-6, doi: 10.1109/ICRITO.2018.87488

[6] Cichocki, Andrzej, and P. H. A. N. Anh-Huy. “Fast local algorithms for large scale nonnegative matrix and tensor factorizations.” IEICE transactions on fundamentals of electronics, communications and computer sciences 92.3: 708-721, 2009.

[7] “Online Learning for Latent Dirichlet Allocation”, Matthew D. Hoffman, David M. Blei, Francis Bach, 2010