Dissertation on

"Flavour Enhanced Food Recommendation"

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

Submitted by:

Sk Saqlain Mustaq	**01FB16ECS388**
Sumair Ahmed Shariff	**01FB16ECS399**
Harshvardhan Surolia	**01FB16ECS408**

Under the guidance of

Internal Guide
Dr. Dinkar Sitaram
Professor,
Head CCBD,
PES University

External Guide
Name of the Guide
Designation,
Company Name

January – May 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

'Flavour Enhanced Food Recommendation'

is a bonafide work carried out by

Sk Saqlain Mustaq	**01FB16ECS388**
Sumair Ahmed Shariff	**01FB16ECS399**
Harshvardhan Surolia	**01FB16ECS408**

In partial fulfilment for the completion of eighth semester project work in the Program of Study Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period Jan. 2020 – May. 2020. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 8th semester academic requirements in respect of project work.

Signature Dr. Dinkar Sitaram Professor, Head CCBD	Signature Dr. Shylaja S S Chairperson	Signature Dr. B K Keshavan Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. _____	_____
2. _____	_____

DECLARATION

We hereby declare that the project entitled **"Flavour Enhanced Food Recommendation"** has been carried out by us under the guidance of Dr. Dinakar Sitaram, Professor and Head CCBD, and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester January – May 2020. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

01FB16ECS388	**Sk Saqlain Mustaq**	**<Signature>**
01FB16ECS399	**Sumair Ahmed Shariff**	**<Signature>**
01FB16ECS408	**Harshvardhan Surolia**	**<Signature>**

ACKNOWLEDGMENT

We would like to express our gratitude to Prof. Dinkar Sitaram, Dept. of Computer Science, PES University, for his continuous guidance and support, assistance and encouragement throughout the development of this project.

We are grateful to the project coordinators, Prof. Samatha R Swamy and Prof. Usha Devi B G for organizing, managing and helping out with the entire process.

We take this opportunity to thank Dr. Shylaja S S, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support we have received from the department. We would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

We are deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro-Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing various opportunities and enlightenment at every step of the way. Finally, this project could not have been completed without the continual support and encouragement we have received from our parents.

ABSTRACT

The objective of this research project is to develop a mechanism to prove that a user prefers a specific dish and dishes similar to the once the user likes may not be because of the commonality of ingredients or the flavours present in the food item but because of the presence of specific flavonoid (chemical compound) present in the dish. In order to prove the above hypothesis, we have proposed a recommendation system that uses these flavonoids present in the dishes as features rather than using ingredients and flavour scores. This also proves that the performance of the recommendation systems specially used in domain related to food can be improved by incorporating granular level features in their model. The project includes techniques used to collect data and pre-process them and also model training and in the end in-order to assert the success of the hypothesis we have compared the result of multiple models .

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Recommendation Systems are an imminent part of almost all sites that exist today, this is because almost all the sites work by presenting data to the user so that the user can interact with it and all the following things can happen, but the problem is that the amount of data that is present is humongous as it cannot be fetched at a single shot or even though if it is fetched at once it becomes very incomprehensible for the user in order to carry out a decision. Thus the role of the Recommendation System is to ease the selection process of the user so that the user need not waste time in searching for the product as the Recommendation System will learn the preference of the user and recommend items.

Food is an intrinsic part of all human beings. With the immense amount of data about food and their nutrient content present today it encourages the development of various food computing methods. Food computing basically involves acquiring data and identifying regions where it can be incorporated thereby giving affirmative results. Food computing collates data from various sources and involves work such as recognition, retrieval, prediction and monitoring of food intake, recommendation. One of the major results of food computing is understanding the correlation between food product choices and health value related to them respectively. In the present world everyone is health conscious and is serious about what they eat as a healthy diet decreases the probability of acquiring any fatal disease. To help in making a healthy diet, algorithms can compute health scores based on the user's physical and medical conditions. However most of the food that falls under the healthy domain lacks taste. Thus the user prefers to eat healthy food by sacrificing the pleasures of taste, and this is what most of the recommendation systems that are being used in food-related sites have incorporated. We have proposed a mechanism where we have implemented a recommendation system that incorporates flavonoid(chemical compounds) present in the food as features thereby not only recommending healthy food but also taking into consideration the taste preference of the user.

Chapter 2

PROBLEM DEFINITION

The goal of the project is to build a recommendation system which incorporates flavonoid(chemical compounds) present in the food as features as most of the food that falls under healthy domain lacks tastes. Thus the user prefers to eat healthy food by sacrificing the pleasures of taste, and this is what most of the recommendation systems that are being used in food-related sites have incorporated. The main objective of the project is to prove the hypothesis that a user prefers similar dishes because there are a specific set of flavonoids that are common among the dishes as the user prefers those specific sets of flavonoid molecules. We try to determine to what extent will the recommendation model improve when it incorporates just the flavonoids compared to the recommendation model which just incorporates the ingredients and the flavour. And in the end we have also tried to make a health score calculator based on the users physiology. Combining the two properties that is, both the user preference of the flavonoids and the health aspect of the user we are not only recommending food that the user prefers but also healthy dishes.

Chapter 3

LITERATURE SURVEY

3.1 Taste-nutrient relationships in commonly consumed foods.

3.1.1 Introduction

The paper proposes an investigative study to determine if there is any association between the nutrients in the food and taste that a person can perceive basically whether taste can act like a nutrient sensor, mainly focused on five basic tastes that are sweetness, saltiness, savouriness, sourness and bitterness.

3.1.2 Implementation

In order to carry out the study the potential subjects were recruited mainly men and women aged between 18-35 years they had certain specific physiological characteristics , subjects who had any allergies specific to certain food items were excluded. Only fifty food items were chosen for the study which were commonly consumed within the Netherlands. The author states that they have used spectrum methods to obtain ratings of the taste intensity for the 5 basic tastes. Subjects evaluated the taste intensities based on a reference solution which contained increasing concentration of sucrose, NaCl, , MSG, citric acid, caffeine separately which mapped to sweetness, saltiness, savouriness, sourness and bitterness respectively. The subject were given solution of varying concentration and rated the taste intensity on a scale from 0 to 15

3.1.3 Evaluation and Results

Simple and multiple regression analysis were performed using PROC REG to test association of taste intensity rating with nutrient concentration , the nutrient concentration was treated as the

independent variable and the intensity ratings were treated as the dependent variable. The author concludes by revealing some positive results by stating that a large part of the sweetness could be a result of the mono and disaccharide content and both saltiness and savouriness were associated with Na and protein content. But the author says that the experiment fails to give any results that shows a correlation between nutrients with bitterness. The paper does not propose a mechanism to calculate a rating score based on the chemical compounds present in the food items, but is successful in giving proper evidence that there is a correlation between the taste and chemical compounds present in the food. Thus this notion can be used to build a recommendation system which we have incorporated in our project.

3.2 Joint Deep Modelling of Users and Items Using Reviews for Recommendation

3.2.1 Introduction

The paper proposes a deep learning based model that could be used for recommending items, the author believes in the notion that most of the useful information exists in the reviews given by users and this major information has been ignored by most of the recommendation systems as it has the capacity to solve the sparsity problem that is commonly encountered in most of the recommendation system and can also improve the quality of the recommended products. Therefore the paper proposes a deep learning based model called DeepCoNN(Deep cooperative Neural Networks) that makes use of the rating as well as the reviews to recommend items.

3.2.2 Implementation

The model consists of two parallel networks coupled in the last layer. One of the layers focuses on learning user behaviours by exploiting reviews written by the users and the other learns item properties from the description of the items. The shared layer enables latent factors learned from users and items to interact with each other in a manner similar to matrix factorization techniques. A significant advantage of the DeepCoNN is that it has the capability to model users and. Items in

a joint manner as most of the other similar algorithms perform the modelling independent of the ratings, therefore there is no guarantee that the learned factor can benefit in proper rating. The model consists of two parallel networks , the first layer is a lookup table followed by a Convolutional neural network which hopefully discovers features of users and items. A top layer is added in the end so that both the networks can interact with each other this allows the hidden latent factors of user and item to interact with each other, this layer calculates the rating predictions. The overall architecture of the system is shown in the below image.

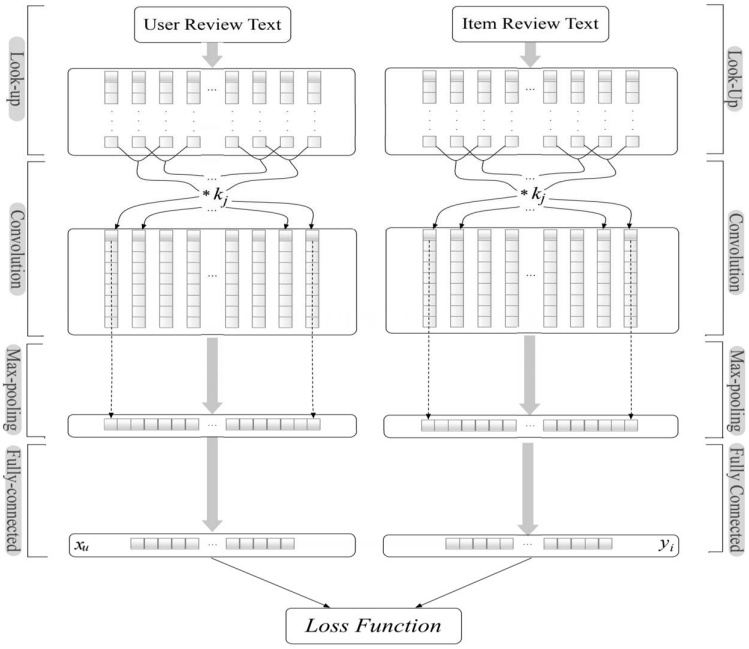

Fig 3.1 DeepCONN Architecture

3.2.3 Evaluation and Results

The model was tested on multiple datasets such as Yelp, amazon, and Beer(ratebeer.com), the table below gives the summary of the dataset. To validate the effectiveness of the model three more algorithm were taken which were similar in action to the current model, Matrix Factorization, Probabilistic Matrix Factorization(PMF), Latent Dirichlet Allocation (LDA) and Collaborative Deep Learning(CDL) all the algorithms except MF and PMF have incorporated reviews. The

model uses MSE metric to evaluate the performance. The performance of the model compared to the other algorithms is listed in the table below.

Class	#users	#items	#review	#words	#reviews per user	#words per review
Yelp	366,715	60,785	1,569,264	198M	4.3	126.41
Amazon	6,643,669	2,441,053	34,686,770	4.053B	5.2	116.67
Beer	40,213	110,419	2,924,127	154M	72.7	52.67

Fig 3.2 Dataset Details

The author states that almost all the recommendation systems suffer from two common problems that are cold start and sparsity. Cold start basically occurs when a new user enters the system and the system fails to recommend because of the lack of any data pertaining to that specific new user or when a new item enters the system , since there no existing customers that have yet opted that specific item it becomes difficult in recommending it. The proposed model also aims to solve the above two problems as has the capability to learn fine grain features from the reviews.

Dataset	MF	PMF	LDA	CTR	HFT-10	HFT-50	CDL	DeepCoNN	Improvement of DeepCoNN (%)
Yelp	1.792	1.783	1.788	1.612	1.583	1.587	1.574	**1.441**	8.5%
Amazon	1.471	1.460	1.459	1.418	1.378	1.383	1.372	**1.268**	7.6%
Beer	0.612	0.527	0.306	0.305	0.303	0.302	0.299	**0.273**	8.7%
Average on all datasets	1.292	1.256	1.184	1.112	1.088	1.09	1.081	**0.994**	8.3%

Fig 3.3 Performance of different model

The DeepCoNN not only resolves the sparsity and the cold start problems but it also has an advantage that for users or items with lower number of ratings , DeepCoNN reduction in MSE is higher. Because of all these properties we have tried to build the recommendation system for our project based on the above proposed architecture and see whether it will further improve the model behaviour in addition to the features that we will be using.

3.3 Automated Healthy Dish Recommendation.

3.3.1 Introduction

One of the most common causes of chronic diseases is the lack of proper healthy dishes in everyday diet. Health is something that is a consequence of our lifestyle. Dietary choices are a major component of our lifestyle. It is very difficult to have a health professional to be beside all the time when making some dietary choices. Recommendations which incorporate expert knowledge are the key to unravel healthy diets in the world. The author proposes a methodology which stiches the two fundamental ideas such as tracking the diet of the user and providing actionable suggestions to improve one's health.

3.3.2 Implementation

The overall architecture of the system involves an algorithm which takes in the user's current location and fetches all the nearby restaurant's menu and calculates a health score based on the nutrient contents present in the dishes and then recommends it to the user. The system architecture has two parts, the first major component includes a data filter to ensure recommended items are of higher quality. Quality is checked by numeric checks on the ingredients and nutritional values. The filters are based on Calories, Carbohydrates, Fat , Red Meat. The second component consists of a personalized health score calculator which simulates an health advisor which makes use of expert knowledge. The author proposes an algorithm called ELIXIR(Environment and Life Integrated eXpert Individualized Recommendation System) which uses expert tuned weights which has been manually adjusted till the rank list of items were as close as those that were listed out by an actual nutritionist, the weights are shown in the below diagram. The algorithm groups the macro nutrients into three different categories, recommended base, additional nutrients and restricted base nutrients. The algorithm places healthy components in the numerator and unhealthy components in the denominator then the ratio is scaled from 1-100. The algorithm that the author proposes is shown in the below figure.

Nutrients	Weight	Daily value
Calories	1.00	2000 cal
Protein	1.00	50 g
Sugar	1.10	50 g
Total Fat	1.10	60 g
Saturated Fat	1.70	20 g
Carbohydrate	1.00	300 g
Fiber	1.50	25 g
Sodium	1.00	2400 mg
Cholesterol	1.20	300 mg
Vit A	1.00	5000 IU
Vit C	1.00	60 mg
Calcium	1.00	1000 mg
Iron	1.00	18 mg
Trans Fat	0.91	NA
Complex Carb	0.10	NA

Fig 3.4 Nutrition value

1 : **procedure** ELIXIR − SCORE (weights, Daily Values, Mult)
2 : $RecBN = (Protien, Fiber)$
3 : $RecAN = (VitA, VitC, Ca, Fe)$
4 : $RestBN = (Cal, Chol, Na, SatFat, TotFat, Sugar)$
5 : $RecBase = \sum_{i \in RecBase} weights\,[i] * \frac{dish[i]}{DailyValues[i]}$
6 : $RecBase = RecBase + weights\,[Fiber] * \frac{dish[Fiber]}{dish[Carb]} + weights\,[ComplexCarb] * \frac{(dish[Carb] - dish[Fiber] - dish[Sugar])}{dish[Carb]}$
7 : $RecAdd = \sum_{i \in RecAN} weights\,[i] * \frac{dish[i]}{DailyValues[i]}$
8 : $RestBase = \sum_{i \in RestBN} weights\,[i] * \frac{dish[i]}{DailyValues[i]}$
9 : $RestBase = RestBase + weights\,[Carb] * \frac{dish[Sugar]}{dish[Carb]} + weights\,[SatFat] * \frac{dish[SatFat]}{dish[TotalFat]} + weights\,[TransFat] * dish\,[TransFat]$
10 : $BaseElixir = \frac{(RecBase + Mult * RecAdd)}{((1+Mult) * (RestBase))}$
11 : **return** $BaseElixir$
12 : **end procedure**

Fig 3. 5 ELIXIR Algorithm

3.3.3 Evaluation and results

In order to validate the model the author has used the USA food consumption database. The author has compared the ELIXIR model with similar models such as FSA,SAIN and NRF, the energy density plot of the mean health score for different food groups reveals that ELIXIR is capable of better segregating health food from unhealthy food. Also validated the model by getting the 50 recommendations evaluated by an expert dietitians in this case also the ELIXIR has an overall advantage where it had scored 8.9 out of 10 the remaining SAIN 8.2, FSA 7.1 and NRF 6.9. Because of the beauty of the algorithm we have tried to replicate this as it would be great if we calculate health score once the dishes have been recommended as a result the user will not only get the recommended food which is in preference to the user but also get healthy food.

3.4 Flavour Enhanced Food Recommendation.

3.4.1 Introduction

The author has proposed a mechanism to incorporate flavour features to enhance the quality of the recommendation system, one of the most common way of incorporating this features into the recommendation system is to make use of the factors that effects the gustatory receptors. If such features have been incorporated into the system there will be an overall advantage of suggesting food items that the user is more likely to enjoy as the recommended food items are in preference to the taste of the user. The author has formulated a technique to calculate scores associated to each flavours such as, salt, sweet, bitter, Umami and Richness. The author states that there are specific set of chemicals that simulates these flavour receptors in our tongues. The Below diagram shows how various different chemical compound can simulate different taste receptors.

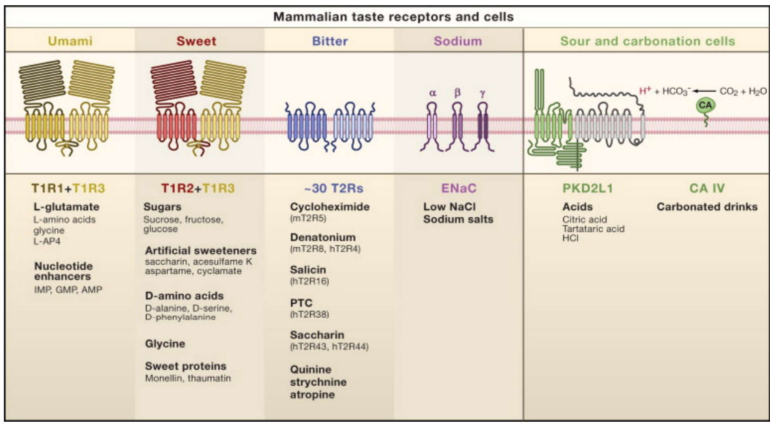

Fig 3.6 Chemical Compounds

3.4.2 Implementation

The overall architecture consists of a database, the flavour profiler and the recommendation system. The database consists of two components ,one contains the item information such as the

ingredients and nutrient contents and the other contains user information such as rating. The summary of the dataset is given in the below diagram. Most the data has been gathered from sites like myfitnesspal, yummly and allrecipes.

$$W_{x,y} = tf_{x,y} \cdot \log\left(\frac{N}{df_x}\right)$$

Where,

- x is the set of tags and y is the set of dishes
- $tf_{x,y}$ = 1 if dish y has tag x else 0
- df_x = number of dishes containing tag
- N = total number of dishes

Fig 3.7
Tagging

User Database, Total reviews	30,193
User Database, Unique Users	22,625
User Database, Users with greater than 5 reviews	466
Food Database, Total Dishes	1381
Food Database, Indian Dishes	1051

Fig 3.8
Dataset used in paper

$$\forall i \in All\ dishes\ in\ the\ database \frac{\sum_{\alpha} \cdot \beta \cdot UserScore(i)}{\sum_{\alpha} \beta}$$

Fig 3.9
Calculating Score

For each food item in the database the five basic flavour intensities were estimated on a scale of 1 to 10. This is done by recognizing the most influential chemical for that specific flavours

respectively present in that dish. Once the flavours were profiled it was used as an additional feature with the ingredients and that constituted the feature vector associated with each dish. The author has implemented a content based filtering algorithm that takes the content of the item such as its properties in this case it would be ingredients and flavour scores. Tf-Idf embedding is used to encode the data the formula is shown in the above figure. In order to predict the rating for a new dish for a given user the formula shown in the image is used, suppose a rating has to be calculated for a new dish j then α is the set of dishes that the user has already rated and β is the cosine similarity between the dish whose rating has to be calculated and the dish that the user has already rated, and in the end the result is normalized over all the cosine similarity calculated.

3.4.3 Evaluation and Results

In order to evaluate the model A/B testing between three various recommendation approaches as shown in the table , it is evident from the result that tfidf along with incorporated flavours enhances the performance of the recommendation system.

Method	RMSE
Matrix Factorisation	2.93
TF-IDF	2.11
TF-IDF with flavour	1.94

Fig 3.10 Performance of recommendation system

Our project is an extension of the above paper , the paper fails to give proper evidence on how the flavour scores are calculated as the weights that are being used to estimate the flavour score is not proved with any proper reference and also the way the evaluation is carried out is also vague. Thus we want to resolve these problems and also instead of just using the ingredients and flavour score as feature vector for dishes we would like to dig deep into this by not just using the ingredients directly but using the flavonoid that is present in the ingredients as a feature for each dish and then train a recommendation system.

Chapter 4

PROJECT REQUIREMENT SPECIFICATION

4.1 Methodology

4.1.1 Proposed Approach

- The Food related data such as ingredients, nutrient content, rating and review was collected by scraping data from Yummly
- The flavonoids data was collected from FlavorDB.
- The ingredient data was converted into respective flavonoid molecules
- Tf-idf was constructed to vectorized the data
- The vectorized data was used to train a content based filtering algorithm and also to train a recommendation system.
- Health score is calculated in the end using ELIXIR algorithm
- The model is evaluated to check for the results.

4.2 Dependencies/Assumptions

- We assume that all the flavonoids data corresponding to all the ingredients are present and are correct as we are collecting it from some other source
- We are only considering the users that have reviewed items more than 5 times, as a result we will have sufficient data to capture user preference and also to overcome the sparsity problem that may arise.

RISK

- As we are working with flavonoid molecules that are chemical compounds present in each specific ingredient ,finding such data is extremely difficult and even if it is found the data should be gathered from trusted sources only.

4.3 Environment Requirements

4.3.1 Hardware Requirements

- As the project requires text processing and we are dealing with reviews and also trying to train a deep learning based model we require moderate computational powers, a minimum of 4 GB ram and a GPU would be preferable.

4.3.2 Software Requirements

- Python 3.0 or greater,
- Windows or linux operating system
- Jupyter notebook
- Google Colab
- Python packages like numpy,pandas,nltk,mechanize,beautifulsoup,sklearn
- Frameworks like Keras,Tensorflow

Chapter 5

SYSTEM DESIGN.

5.1 High Level System Architecture

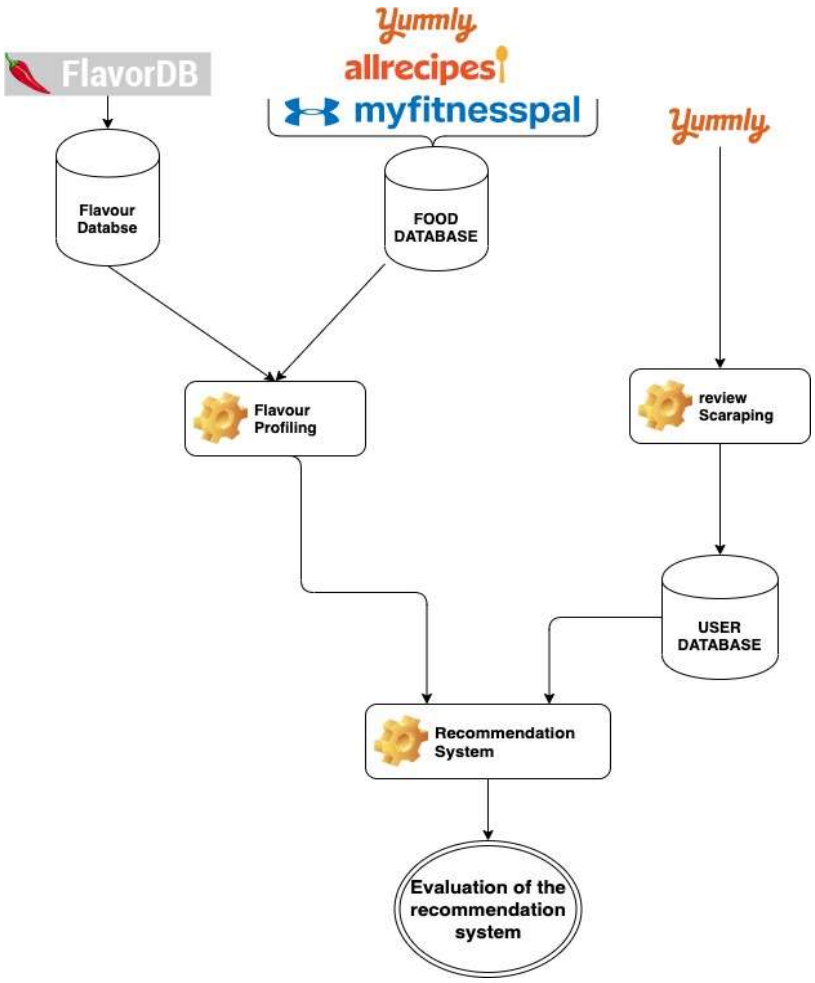

Fig 5.1 Architecture of the system

The above figure describes the overall architecture of our system

5.2 Detailed Design.

The entire architecture can be broken down into 4 parts – data collection, pre-processing and flavour profiling, building the recommendation system and then evaluation of the recommendation system.

5.2.1 Data Collection.

- In order to build the recommendation system we needed user data and food data.
- The data related to the dishes such as the nutrient contents, ingredients were collected by scraping it from Yummly, MyFitnessPal and allrecipes.
- The data related to users such as the ratings were collected from Yummly , but we needed reviews related to each dish also as we had planned to train a deep learning based model that needed user review data.
- Initially the user review data was collected from twitter using twitter Api , the pseudo is shown below, tweets were collected for specific hashtags that's nothing but the dish name that we had in our database .

```
def getTweets(foodName="",cnt=100,pg=100,startDate='2018-01-01',endDate='2020-01-02'):
    if(len(foodName)>0):
        #tweetEntry=[]
        for page in tweepy.Cursor(api.search,q=foodName,count=cnt,include_rts=False,since=startDate,tweet_mode='extended',lang='en').pages(pg):
            with open(foodName+".json","w") as fr:
                for status in page:
                    status=status._json
                    tweet=status
                    #tweetEntry.append([foodName,tweet["user"]["id"],tweet["full_text"]])
                    json.dump([foodName,tweet["user"]["id"],tweet["full_text"]],fr,indent=2)

    else:
        print("invalid food name")


dishFile=open("dishName.txt","r")
dishRaw=dishFile.readlines()

print(len(dishRaw))
for i,ele in enumerate(dishRaw):

    if(i>0):
        try:
            print(i)
            getTweets(foodName=ele.strip("\n"),cnt=100,pg=100)
        except:
            time.sleep(15*60)
            print(i)
            getTweets(foodName=ele.strip("\n"),cnt=100,pg=100)
```

Fig 5.2 Code to collect data from twitter

- But the twitter data did not have any ratings associated with them so we used NLTK Vader library, which analysed the sentiment of each review and produces a rating in the range of 1-5. The reason we used Vader library is because it proved to work really well on social media data. It does a rule based analysis on the input text and throws out a score.
- There were some inherent problems with the above approach of data collection ,firstly there was no way of validating the method that we used to assign rating to each review and second the data that we had collected was not proper about 70% of the users had given reviews only once , and the reviews that were given by most of the users were not related to food at all , the below figure summarises the problem that existed in our data.

```
Total number of users =  27442
Total number of users with 1 review =  19382 =>  70.62896290357845 %
Average number of times that a user has rated =  1.2465137406313878
```

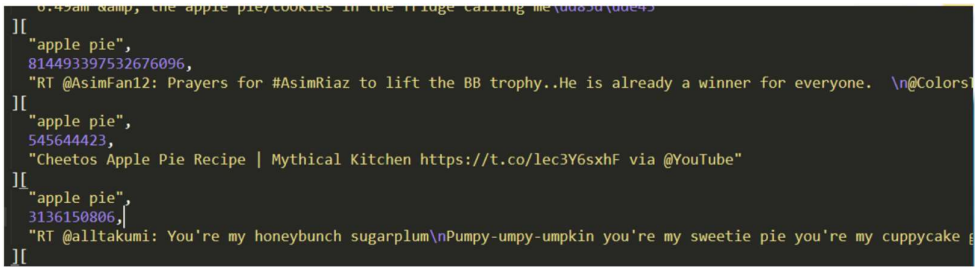

Fig 5.3 Snapshot of the data collected

- In order to overcome the drawbacks of the previously mentioned dataset, we managed to scrape legitimate food review data from Yummly by making use of beautiful soup and mechanize library as Yummly developer API's were no longer available. The below snippet of code shows how we scrape data from Yummly. A delay of 0.5 second was added between each query to simulate a normal behaviour when an user interacts with the website.

```
def getFoodReviews(food):
    food=food.replace(" ","+")
    url="https://www.yummly.com/recipes?q="+food+"&taste-pref-appended=true"
    response=br.open(url)
    soup = BeautifulSoup(response.read(), 'html.parser')
    recipeContainer=soup.find_all("div",class_="RecipeContainer")
    foodContainer=recipeContainer[0].find_all("a")
    print(len(foodContainer))
    reviews=[]
    id=1
    for i in range(0,len(foodContainer),2):
        foodCode=foodContainer[i]["href"]
        if(not foodCode.startswith("/recipe/")):
            continue
        url="https://www.yummly.com"+foodCode
        response=br.open(url)
        soup=BeautifulSoup(response.read(),'html.parser')
        cookbookDetails=soup.find(class_="recipe-details")

        reviewUrl="https://www.yummly.com"+foodCode+"#reviews"
        response=br.open(reviewUrl)
        soup=BeautifulSoup(response.read(),'html.parser')
        allReviews=soup.find_all("div",class_="review media")
        if(len(allReviews)==0):
            continue

        for eachReview in allReviews:
            body=dict()
            body["id"]=id
            body["food_item"]=food
            body["name"]=eachReview.find(class_="review-name").find("a").get_text()
            stars=eachReview.find(class_="review-rating").find_all("span")
            cnt=0
            for st in stars:
                if(st["class"][1].startswith("full-star")):
                    cnt+=1
            body["rating"]=cnt
            body["review"]=eachReview.find(class_="review-text font-normal p2-text").get_text()
            reviews.append(body)
        id+=1
        time.sleep(0.5)
    return reviews
```

Fig 5.4 Code to collect reviews from Yummly

- Since we are aiming to incorporate flavour features into recommendation, we are using the chemical compounds also known as flavonoid molecules present in the food as features rather than the ingredients and flavour scores this is because we believe that the user prefers similar food items because of the presence of specific flavonoids that might be common among the dishes. In this way we are able to construct features at a much more micro level so that we might be able to elevate the performance of the recommendation model.
- In order to collect the flavonoid information we used FlavorDB. It is a product of Complex Systems Lab, IIIT-Delhi. The database comprises 25595 flavour molecules. We have written a script like the one to scrape data from Yummly to also scrape flavonoids data from FlavorDB. The code snippet shown below demonstrates how we

scraped data. Totally we have somewhere around 100 unique ingredients and after combining all the flavonoids present in each ingredient we totally have around 1405 unique flavonoid molecules.

```
def getChemCompAndFlavours(ingredient):
    #initializing mechanize
    br=mechanize.Browser()
    #loading the page
    response=br.open("https://cosylab.iiitd.edu.in/flavordb/search")
    br.response()
    #extracting the required form to search for chemical compound
    br.form=list(br.forms())[1]
    #filling the form with the required ingredient
    control=br.form.find_control("entity")
    control.value=ingredient
    #making request
    response=br.submit()
    #reading response in JSON format
    rawJsonData=response.read()
    #selecting a specific ingredient from the list of all possible similar ingredients
    entity_id=None
    jsonifiedData=eval(json.loads(rawJsonData))
    #print(jsonifiedData)
    for ele in jsonifiedData:
        if(ele["entity_alias"].lower()==ingredient.lower()):
            entity_id=ele["entity_id"]
            break
    if(entity_id==None):
        try:
            entity_id=jsonifiedData[0]["entity_id"]
        except:
            return list()
    print(entity_id)
    queryString="/flavordb/entity_details?id="+str(entity_id)
    print(queryString)
    url="https://cosylab.iiitd.edu.in/flavordb/search"+"/"+queryString
    print(url)

    #loading the chemical compound page
    response=br.open(url)
    soup=BeautifulSoup(response.read(),'html.parser')
    table=soup.find_all('table',id="molecules")
    table[0].findAll("tr")
    molecules=dict()
    for tr in table[0].findAll("tr"):
        td=tr.findAll("td")
        if(len(td)==4):
            chemComp=td[0].get_text(strip=True)
            molecules[chemComp]=td[2].get_text(strip=True).split(",")
    return molecules
```

Fig 5.5 Code to scrape flavonoids

- The table below summarizes everything.

User Database, Total Reviews	106511
User Database, Unique Users	21004

Food Database, Total Dishes	1381
Food Database, Unique Flavonoids	1405

Table 5.1 Summarization of dataset

5.2.2 Data Pre-Processing

- The raw food data that was collected by scraping Yummly, was stored in json format , later based on pattern matching ingredients were extracted. The figure shows how the data is pre-processed; only the ingredients are extracted as a result we ended up with around 100 unique ingredients that is distributed among 1381 dishes.

Fig 5.6 Data pre-processing

- Since we are using flavonoid molecules in our recommendation system. We have used FlavorDB to get the flavonoids associated with each unique ingredient. The below figure illustrates how we have extracted flavonoids related to a specific dish.

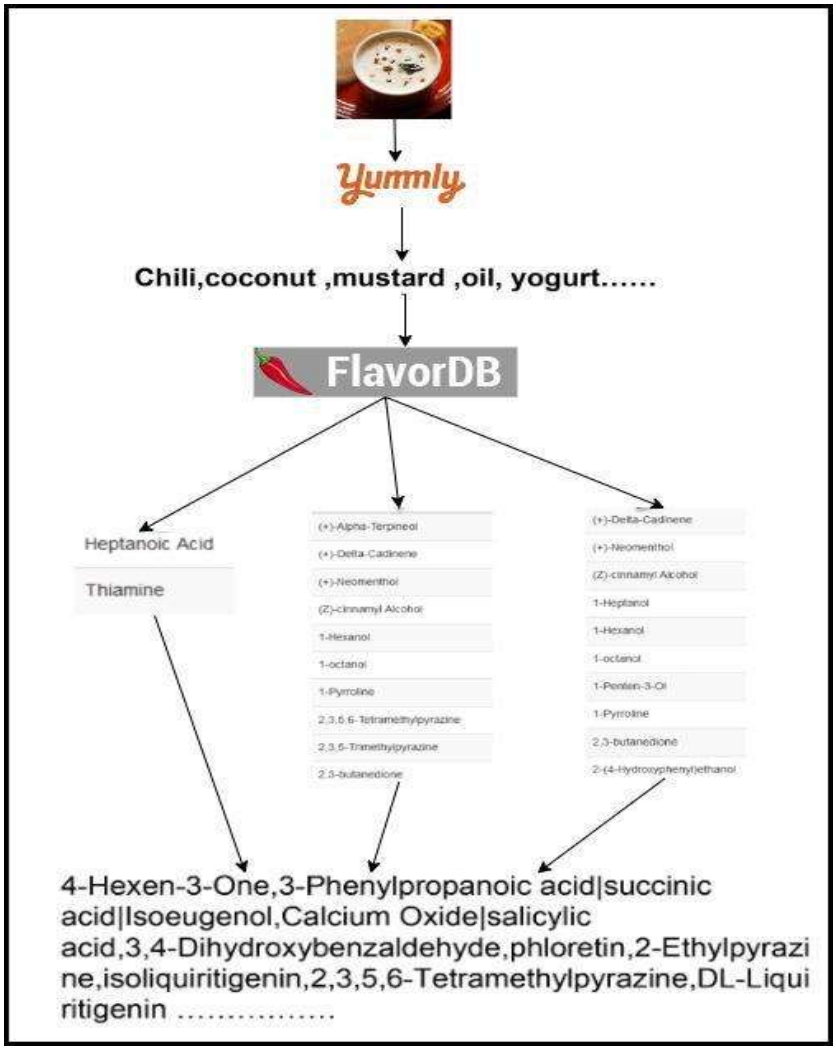

Fig 5.7 Flavonoids for Coconut Chutney

- The user data such as the review and ratings were stored in json format. Before the data was used any further in the recommendation system , the reviews were processed , that is normal text processing was done such as tokenization, removing stop-words, special characters and then lemmatization was done.

5.2.3 Recommendation System.

We have implemented two recommendation systems, the former uses content based filtering algorithm and the latter uses deep learning based algorithm called DeepCoNN (Deep Cooperative Neural Network).

5.2.4 Evaluation.

In-order to evaluate the performance of the system we have used MSE as our error measure to compare various models. We will also carry out a survey to check how well our proposed model works.

Chapter 6

RECOMMENDATION SYSTEM

In order to prove our hypothesis that incorporating micro level details(flavonoid) in our features will enhance the performance of the recommendation system or not, we have implemented two recommendation systems, the former uses content based filtering algorithm and the latter uses deep learning based algorithm called DeepCoNN. In-order to compare the models performance we have used the content based filtering algorithm which has been trained with only food ingredients and flavour score.

6.1 Content Based Filtering algorithm.

6.1.1 Introduction

The reason that a content based filter algorithm is used is because, in addition to the rating it takes the information about the item into consideration while making recommendations. On the Other hand Collaborative filtering just takes the user ratings into consideration and when it makes any recommendation for a user, it takes the current users rating history and does a similarity among the remaining users to find which users are most similar to the current user and thereby recommending dishes that the other user have preferred, basically the notion behind collaborative filtering is that users who view and evaluate items in a similar way are more likely to take up dishes in a similar manner.

The other method of recommending dishes would be to use the Matrix Factorization method which falls under collaborative filtering algorithm also. The model takes in a user-item sparse matrix and tries to fill the sparse content with some positive rating; this is done by decomposing the matrix into two matrices that would constitute an item matrix and a user matrix. The model tries to fill in the sparse cells such that the original contents in the matrix is retrained as close as possible

simultaneously filling in the remaining cells. The figure below illustrates the methodology. The problem with the collaborative filtering based algorithms is that it is

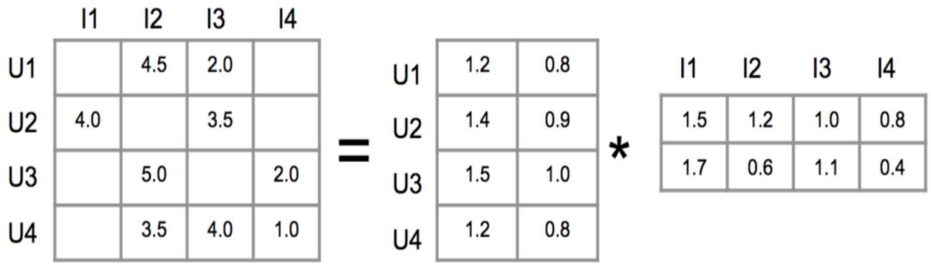

Fig 6.1 Example of matrix factorisation

computationally expensive and also when making any recommendation for any user it only considers other similar users and also it does not take into consideration the contents of the item hence the flavonoids that we have gathered cannot be substituted or taken into consideration while making any recommendation.

6.1.2 Implementation

In order to build the content based filtering , we had to perform some pre-processing that is convert all the ingredients into their respective flavonoid contents and after that process the flavonoid data that is convert it from text to some numerical value, we could not use any embedding techniques because there wasn't any embedding available for just chemical compounds. So we used TF-IDF (Term Frequency Inverse Document Frequency) technique as it is famously used in many information retrieval techniques and it is also proved to enhance the models performance that deals with text data. Each dish in our database has got a tag associated with it as shown in the figure below.

1	Ethyl Lactate|3,4-Dihydroxybenzaldehyde|DL-Liquiri
2	AC1LDI49|56424-87-4|3,4-Dihydroxybenzaldehyde|
3	3-Methyl-1-butanol|Thymol|2-Nonanone|Pyrrolidine|

Fig 6.2 Tags associated with the dishes

We have associated all our 1381 unique dishes with 1405 unique flavonoid molecules. Then for each tag associated with a dish ,TF-IDF was calculated using the formula given below.

$$W_{x,y} = tf_{x,y} \cdot \log\left(\frac{N}{df_x}\right)$$

Where,

- x is the set of tags and y is the set of dishes
- $tf_{x,y}$ = 1 if dish y has tag x else 0
- df_x = number of dishes containing tag
- N = total number of dishes

Fig 6.3 Tf-IDF

Once the TF-IDF scores are calculated for each dish , when a recommendation has to be made the below formula is used to calculate the score for a new dish j using the scores of all the dishes i that is rated by the user a weighted average is calculated, the weight β is nothing but the cosine similarity between the dish j and dish i , in this way a new preference score is calculated. The below snippet of code shows how we used the formula to estimate a rating for a new dish for a specific user

$$\forall i \in All\ dishes\ in\ the\ database \frac{\sum_\alpha \cdot \beta \cdot UserScore(i)}{\sum_\alpha \beta}$$

```
def make_predictions(db, ratings_train, ratings_test, include_flavours):
    """
    Using the ratings in ratings_train is used to make prediction
    This is done by computing the weighted average
    rating for every other dish that the user has rated.
    """
    result = []
    x = 0
    for index,row in ratings_test.iterrows():
        # mlist contains dishIds rated by the user in the train set
        mlist = list(ratings_train.loc[ratings_train['userId'] == row['userId']]['dishId'])
        # csr list contains tfidf scores of tags for dishes rated by the user
        csrlist = list(db.loc[db['dishId'].isin(mlist)]['features'])
        # mrlist contains scores of dishes rated by the user (dishes in mlist)
        mrlist = list(ratings_train.loc[ratings_train['userId'] == row['userId']]['rating'])
        # computing similarity between dishes user rated and the current dish in the test set
        sim = [cosine_sim(c,db.loc[db['dishId'] ==row['dishId']]['features'].values[0], include_flavours) for c in csrlist]
        # computing similarity times the rating for known dish
        wan = sum([ v*mrlist[i] for i,v in enumerate(sim) if v>0])
        wadlist = [i for i in sim if i>0]
        ## check for sum(wadlist) > 1
        if len(wadlist)>0 and sum(wadlist) >= 1:
            result.append(wan/sum(wadlist))
            x = x + 1
        else:
            # if dish did not match with anything approx as average of users rating
            result.append(np.mean(mrlist))
    return np.array(result)
```

Fig 6.4 Code to make recommendation to the user

The same method is also used as a baseline to compare various models that we have implemented, the difference is that the baseline model is just trained over ingredients. The result and the comparison with the baseline model is discussed in the next chapter.

6.2 DeepCoNN(Deep Cooperative Neural Network)

6.2.1 Introduction

One of the major drawbacks of the Collaborative filtering as also discussed in the above sections is that they model users and items just based on the scores provided by the user and ignore the most vital part that is the review given by the users. The same problem is faced by the content

based filtering technique as it only captures the rating and item description , it fails to model the user and item description as a whole. This basic problem is what the DeepCoNN is trying to solve.

The DeepCoNN is a Neural Network that tries to model users and items in a jointly manner thereby making use of the review text to predict item ratings. The architecture of the neural network is shown below. The DeepCoNN also tries to solve the most common problem that is encountered in the recommendation system that is the cold start problem and the sparsity problem. Cold start problem is nothing but the situation that arises when a new item or a user enters the system and the sparsity problem arises when a user has rated only a few items as a result there's no sufficient information present about the user to make any recommendation. Thus the DeepCoNN has proved to solve the above stated problems and also proven to elevate the performance of the recommendation system.

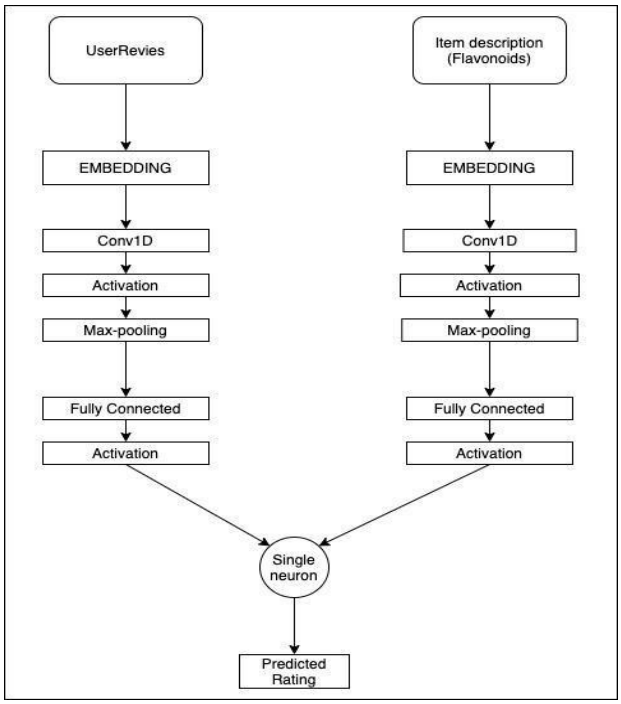

Fig 6.5 DeepConn

6.2.2 Implementation

Before implementing the model the data had to be processed, that is for every review that was given by the user we had to associate the flavonoid related to that dish the user had rated , the figure below illustrated the data format that was needed by the model.

user_id	food_id	rating	userReviews	foodReviews
5169	745	1	['did', 'not', 'taste', 'good', 'at', 'all']	['AC1LDI49', '56424-87-4', '3,4-Dihydroxybenza...

Fig 6.6 Preprocessed data

Once the appropriate data format was obtained, further processing was required that is converting the text to some numerical data. Since one side is natural language and the other side is a list of flavonoids it is not possible to embed the data as almost all embedding algorithms are trained on natural languages and its constructs. So the only one possible way that we could find of encoding the data into numerical format was to use TF-IDF, as we used in the above algorithms also. In the end we had around 6165 unique tokens for reviews and 1405 unique tokens for flavonoids which matched with the exact count that we had in our database.

Once the data processing part was completed, the architecture that was given in the DeepCoNN paper, also as shown in the above figure was replicated initially. The first layer consists of a lookup table, then the input was made to pass through an embedding layer, the embedding layer converts the fractions into vectors which was further needed by the next layers. The next in line is an 1-D convolutional layer which performs feature extraction, next comes the activation function, then max-pooling , then the data is flattened and made to pass through a dense layer. The same architecture is replicated for both the user and item network. In the end the output of both the dense layers are merged in-order to extract information from both the item description and the item review and then the output of the merged layer is made to pass through a single neuron to get the predicted rating. MSE is used as an error measure. The below snippet of code, shows how the

architecture is implemented, as it can be seen that 5% of the training data is used as validation set while training the model. The output of the model and its comparison to the other models has been discussed in the next chapter.

```
class DeepCoNN():
    def __init__(self,
                 userEmbeddingSize,
                 foodEmbeddingSize,
                 hiddenSize,
                 filters,
                 kernelSize,
                 strides):
        self.userEmbeddingSize = userEmbeddingSize
        self.foodEmbeddingSize=foodEmbeddingSize
        self.hiddenSize = hiddenSize
        self.filters = filters
        self.kernelSize = kernelSize
        self.inputU, self.towerU = self.createDeepCoNnTower(self.userEmbeddingSize)
        self.inputF, self.towerF = self.createDeepCoNnTower(self.foodEmbeddingSize)
        self.joined = Concatenate()([self.towerU, self.towerF])
        self.outNeuron = Dense(1)(self.joined)

    def createDeepCoNnTower(self,embeddingSize):
        inputLayer = Input(shape=(embeddingSize,))
        embeddingLayer=Embedding(embeddingSize,100)(inputLayer)
        tower = Conv1D(filters=self.filters,
                       kernel_size=self.kernelSize,
                       activation="tanh",kernel_initializer='random_uniform')(embeddingLayer)
        tower = MaxPooling1D()(tower)
        tower = Flatten()(tower)
        tower = Dense(self.hiddenSize, activation="relu",kernel_initializer='random_uniform')(tower)
        return inputLayer, tower

    def createDeepCoNnDp(self):
        dotproduct = Dot(axes=1)([self.towerU, self.towerF])
        output = Add()([self.outNeuron, dotproduct])
        self.model = Model(inputs=[self.inputU, self.inputF], outputs=[output])
        self.model.compile(optimizer='Adam', loss='mse')

    def train(self, trainData,trainUserReviews,trainFoodReviews, batch_size, epochs=1):
        tensorboard = TensorBoard(log_dir="tf_logs/{}".format(pd.Timestamp(int(time()), unit="s")))
        self.createDeepCoNnDp()
        userReviews=trainUserReviews
        foodReviews=trainFoodReviews

        self.train_inputs = [userReviews, foodReviews]
        self.train_outputs = trainData.loc[:, "rating"]

        self.history = self.model.fit(self.train_inputs,
                                       self.train_outputs,
                                       callbacks=[tensorboard],
                                       validation_split=0.05,
                                       batch_size=batch_size,
                                       epochs=epochs)
```

Fig 6.7 Code for DeepConn

6.3 ELIXIR

6.3.1 Introduction

Environment and Life Integrated eXpert Individualized Recommendation System also known as ELIXIR is an algorithm that aims at incorporating expert knowledge while making any recommendation. The algorithm has proven to successfully segregate healthy dishes from unhealthy dishes. We have tried to implement the algorithm that is used to calculate the health score with respect to each individual dish.

6.3.2 Implementation

Initially, based on the physiology of the user such as weight, height, gender, age, steps, bmi ratio, we are calculating weights associated with each nutrient, and these weights are specific to a user which is in turn used to calculate health score, the below code snippet demonstrates the process.

```
def adaptive_daily_value(self,weight, height,
                gender, age, steps, height_travelled, bmratio):
    latlong = self.get_lat_long()
    temp = self.get_temp(*latlong)
    altitude =self.elevation(*latlong)

    allowed = {'calories' : 2079.35, 'protein' : 50, 'fat' : 70, 'sat_fat' : 24,
               'carbs' : 310, 'sugar' : 30, 'dietary_fiber' : 30, 'sodium' : 2.3,
               'cholesterol' : 300, 'vitamin_a' : 0.0008, 'vitamin_c' : 0.08,
               'iron' : 0.0087, 'calcium' : 1}

    work = 9.8 * weight * height_travelled * 0.000239006 + weight * steps / 6000

    bmr = weight * 10 + 6.25 * height - 5 * age
    if gender == 'm':
      bmr += 5
    else:
      bmr -= 161

    daily_cal = round(bmratio * bmr + work, 2)
    if temp != None:
      daily_cal = daily_cal * (1 + (85 - temp) / 800)

      for i in allowed:
        allowed[i] = allowed[i] * daily_cal / 2079.35

    na_multi = 1 + 0.015 * (((temp - 32) * 0.56) - 23)
    allowed['sodium'] = na_multi * allowed['sodium'] + (altitude / 1000) ** 2.5

    return allowed
```

Fig 6.8 Code to calculate allowed weights per nutrients

Once the allowed weights per nutrient is calculated specific to the user then that information is used to calculate health score. The nutrients are divided into three categories, Base Nutrients (Protein, Fiber), Additional Nutrients (VitA, VitC, Ca, Fe) and in the end Restricted Nutrients (Cal, Chol, Na, SatFal, TotFat, Sugar). Overall health score is the ratio of the healthy nutrient to the unhealthy nutrients in the dish. The snippet of code shown in the below figure demonstrates how we calculated the health score associated with each food and also specific to each users physique.

```
def elixir(self,allowed, weights, bmratio):

    #recommendaiton base nutrient / positive nutients
    recbn = ['protein', 'dietary_fiber']
    recbase = 0
    for i in recbn:
      recbase += weights[i] * nutrients[i] / allowed[i]
    part1 = 0
    part2 = 0
    if nutrients['carbs']:
      part1 = weights['dietary_fiber'] * nutrients['dietary_fiber'] / nutrients['carbs']
      part2 =  0.1 * (nutrients['carbs'] - nutrients['dietary_fiber'] - nutrients['sugar']) / nutrients['carbs']
    recbase = recbase + part1 + part2

    #restricted base nutrient / negative nutrients
    restbn = ['carbs', 'cholesterol', 'sodium', 'sat_fat', 'fat', 'sugar']
    restbase = 0
    for i in restbn:
      restbase += weights[i] * nutrients[i] / allowed[i]
    part1 = 0
    part2 = 0
    if nutrients['sugar'] and nutrients['carbs']:
      part1 = weights['carbs'] * nutrients['sugar'] / nutrients['carbs']
    if nutrients['sat_fat'] and nutrients['fat']:
      part2 = weights['sat_fat'] * nutrients['sat_fat'] / nutrients['fat']
    restbase = part1 + part2

    #additional nutrients / positive nutrients
    recan = ['vitamin_a', 'vitamin_c', 'calcium', 'iron']
    recadd = 0
    for i in recan:
      recadd += weights[i] * nutrients[i] / allowed[i]
    #score estimation
    mult = bmratio
    div = ((1 + mult) * restbase)
    if div:
      score = recbase + (mult * recadd) / div
    else:
      score = 0

    return score
```

Fig 6.9 Code to calculate health score

The health scores calculated will act like a second pass filter that is once the recommendation system recommends dishes that are in preference to the user flavonoid choices , the recommended dishes might lack health factors associated with it , since we are aiming at recommending not only food that the users prefers mostly but also healthy food. This is where the ELIXIR comes into the picture, where it will take in the users bio information and assign a health score to each recommended dish. Thus it is then up to the user to choose which dish would they prefer.

Chapter 7

TESTING AND RESULTS

Testing the model is one of the most important steps of any research project. It basically acts as a proof to showcase that the proposed hypothesis actually works.
In order to test our model we have around 100K user reviews with 21K unique users, and around 1381 unique dishes with 1405 unique flavonoid molecules.

7.1 Content Based Filtering Algorithm

In order to test the model the dataset was split (75-25) into training and testing data. Two variants of the model were trained ,former being a baseline model which only had the ingredients and the flavour score as features and the latter having the flavonoids data. Mean Square Error metric was used as an error measure.

The baseline model gave an MSE of 1.04 and the model that used Flavonoids gave an MSE of 0.45, which is significantly lower than the baseline. Thus this somewhat proves that incorporating micro level details of a dish such as flavonoid molecules will enhance the performance of the model. In order to further test it A/B testing was done.
The table below summarises all the models.

Method	MSE
TF-IDF with flavours and Ingredients	1.04
TF-IDF with Flavonoids	0.45

Table 7.1 MSE Values

7.2 DeepCoNN

The DeepCoNN model was trained with different configurations and the summary of the result is shown below the table. It can be seen that even with the DeepCoNN network the model performs well. But the. Drawback of this system is that it needs reviews whenever it is making any recommendation thus this becomes a bottleneck in the model.

Model Configuration	MSE
1 convolutional layer with Max Pooling	0.49
1 convolutional layer with Average Pooling	0.9059
2 convolutional layers with Max Pooling	1.2861
2 convolutional layers with Average Pooling	1.4721

Table 7.2 Different model configuration and their results

7.3 Working Demonstration

In-order to demonstrate how the system takes the flavonoid preference into consideration we have taken a user from our dataset, the figure below lists the dishes that are already rated by the user. Once this information was fed into our model, the fig below to the right shows the list of recommended dishes , analysis could be made that the user prefers dessert dishes and the model succeeds in recommending dessert food. Along with the predicted rating the model also calculates health score using the ELIXIR algorithm associated with each recommended dish which is calculated based on the attributes of the user in this case height=5.6 ft, weight=60kg, age=25 ,gender=male , condition=normal ,steps taken=3000, floors covered=10,bmi ratio=22.1.

	dishId	userId	rating	ingredient
0	276	100	5	mango lassi
1	15	100	5	chicken curry
2	176	100	4	nariyal burfi
3	16	100	5	chicken makhani
4	150	100	5	strawberry lassi

Fig 7.1 Dishes Rated by User

dishId	rating	dishName	score
3	4.543339	paratha	0.239879
892	4.523092	chocolate milkshake	0.121095
1373	4.550255	waffles	0.101368
884	4.548975	chocolate brownie	0.097573
891	4.525108	chocolate cheesecake	0.072577

Fig 7.2 Recommended Dishes

Chapter 8

CONCLUSION

In this project we have proposed an hypothesis that replacing the ingredients present in the dish with their flavonoid contents will enhance the performance of a food recommendation system. The experiments carried out that is when the models were implemented and tested, it actually proved in the end that incorporating this micro level details in the model will enhance the recommendation system. We have used filters in the end in order to recommend dishes, the former takes the users taste preference into consideration and then makes recommendations then the health score is calculated using the ELIXIR algorithm and thereby making the recommended dishes not only in preference with the taste of the user but also healthy.

Chapter 9

FURTHER ENHANCEMENTS

1.In this project we have used TF-IDF as embedding for both the text and the flavonoid list for the DeepCoNN model. There's no common embedding that encodes both text and chemical compounds. So in the future if there exists some embedding that is generic in nature then it might be substituted for the TF-IDF which might further enhance the performance of the system

2.In the content based filtering algorithm, in order to extract similar dishes it uses cosine similarity as a metric, since we are using TF-IDF inorder to vectorize the flavonoids, most of the contents in the matrix are empty thus resulting in a huge sparse matrix. If in the future there exists embedding for just chemical compounds then it might be substituted for TF-IDF , if chemical embedding is used then similar flavonoids might be present in the same vector space and in nearby vicinity thereby using the cosine metric in this case makes a much better sense and also the results might improve.

Chapter 10

REFERENCE/BIBLIOGRAPHY

[1] van Dongen, Mirre Viskaal, Marjolijn C. van den Berg, Nicole Vink, Frans J. Kok, and Cees de Graaf. "Taste–nutrient relationships in commonly consumed foods." *British Journal of Nutrition* 108, no. 1 (2012): 140-147.

[2] Zheng, Lei, Vahid Noroozi, and Philip S. Yu. "Joint deep modeling of users and items using reviews for recommendation." In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pp. 425-434. 2017.

[3] Nag, Nitish, Vaibhav Pandey, Abhisaar Sharma, Jonathan Lam, Runyi Wang, and Ramesh Jain. "Pocket dietitian: Automated healthy dish recommendations by location." In *International Conference on Image Analysis and Processing*, pp. 444-452. Springer, Cham, 2017

[4] Nag, Nitish, Aditya Narendra Rao, Akash Kulhalli, Kushal Samir Mehta, Nishant Bhattacharya, Pratul Ramkumar, Aditya Bharadwaj, Dinkar Sitaram, and Ramesh Jain. "Flavour Enhanced Food Recommendation." In *Proceedings of the 5th International Workshop on Multimedia Assisted Dietary Management*, pp. 60-66. 2019.

[5] Yang, Longqi, Cheng-Kang Hsieh, Hongjian Yang, John P. Pollak, Nicola Dell, Serge Belongie, Curtis Cole, and Deborah Estrin. "Yum-me: a personalized nutrient-based meal recommender system." *ACM Transactions on Information Systems (TOIS)* 36, no. 1 (2017): 1-31

[6] Rendle, Steffen. "Factorization machines with libfm." *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, no. 3 (2012): 1-22

[7] Park, Seung-Taek, and Wei Chu. "Pairwise preference regression for cold-start recommendation." In *Proceedings of the third ACM conference on Recommender systems*, pp. 21-28. 2009

[8] Ramos, Juan. "Using tf-idf to determine word relevance in document queries." In *Proceedings of the first instructional conference on machine learning*, vol. 242, pp. 133-142. 2003

[9] FlavorDB- https://cosylab.iiitd.edu.in/flavordb/

[10]Yummly- https://www.yummly.com/

APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS.

CNN	Convolutional Neural Network
NN	Neural Network
DeepConn	Deep Cooperative Neural Network
MF	Matrix Factorization
ELIXIR	Environment Life Integrated eXpert Individualize Recommendation.
TF-IDF	Term frequency – Inverse Document Frequency
RecBN	Recommended Base Nutrients
RecAN	Recommended Additional Nutrients
ResBN	Restricted Base Nutrients