FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Knowledge Graph-Based Recipe Recommendation System

Ricardo Manuel Gonçalves da Silva

DISSERTATION

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Prof. Ana Paula Rocha

January 28, 2020

Knowledge Graph-Based Recipe Recommendation System

Ricardo Manuel Gonçalves da Silva

Mestrado Integrado em Engenharia Informática e Computação

January 28, 2020

Abstract

Recommendation systems are in many and diverse end-user applications, like e-commerce services, online audio or video-sharing platforms as well as media-services and web-content providers. A recommendation system is composed of software tools and techniques whose primary goal is to provide suggestions for items or services a user may wish to utilise.

Regarding the methodology used, recommendation systems are generally categorised as content-based filtering, if the recommendation uses the characteristics of items; collaborative filtering, if a recommendation resorts to the preferences of similar users; and hybrid recommendation, if a recommendation combines the two previous techniques.

Most of the existing recommendation systems use a hybrid approach on the culinary's domain, mixing collaborative filtering and content filtering, focusing on the recipe and the user's rating, behaviours, history, reviews, and preferences. However, culinary's recipes are items formed by heterogeneous information whose content has high value for the user. Good examples are the type of ingredients, ingredient's quantities, cooking procedure, number of steps or recipe's categories. Recent research concludes that models can be improved in accuracy and diversity if the recommendation system considers an item as a composition of characteristics, instead of a singular item approach.

The paper presents an implementation and results of a recipe recommendation system within the culinary's domain, which recognises a recipe as a composition of potential features organised in a structure such as a knowledge graph. This endeavour's resource is a dataset obtained from *Food* website with recipes information and item's explicit rating given by the user. The methodology consists of results comparison between the most common approaches in recommendation systems and knowledge graph-based convolutional network. Graph convolutional network is the only one in our research which produces a model based on a knowledge graph constructed. The experimental results demonstrate an improvement in rating prediction accuracy when using a recommendation system's knowledge graph.

Keywords: Information systems, Information retrieval, Retrieval tasks and goals, Recommendation systems, Knowledge graphs, Graph Convolutional Network.

Resumo

Os sistemas de recomendação encontram-se em diversas e variadas aplicações web, como serviços de comércio eletrónico, plataformas de partilha de áudio ou vídeo online, bem como serviços de "media" e distribuidores de conteúdo. Um sistema de recomendação é composto por ferramentas e técnicas de software cujo objetivo principal é fornecer sugestões para itens ou serviços que um utilizador pode desejar utilizar.

Em relação à metodologia utilizada, os sistemas de recomendação são geralmente categorizados em filtragem baseada em conteúdo, se o sistema de recomendação usar as características dos itens; filtragem colaborativa, se o sistema de recomendação recorrer às preferências de utilizadores semelhantes; e recomendação híbrida, se o sistema de recomendação combinar as duas técnicas anteriores.

A maioria dos sistemas de recomendação existentes usa uma abordagem híbrida no domínio da culinária, combinando filtragem colaborativa e recursos de filtragem de conteúdo, com foco na receita e respetiva classificação, comportamentos, histórico, avaliações e preferências do utilizador. No entanto, as receitas culinárias são itens formados por informações heterogéneas cujo conteúdo tem alto valor para o utilizador. Bons exemplos são os tipos de ingredientes, quantidades de ingredientes, procedimentos de preparação, número de etapas ou categorias de receitas. Pesquisas recentes concluem que os modelos podem ser aprimorados em precisão e diversidade se o sistema de recomendação considerar um item como uma composição de características, em vez de uma abordagem de item único.

Este artigo apresenta uma implementação e os resultados de um sistema de recomendação de receitas dentro do domínio da culinária que considera uma receita como uma composição de recursos potenciais, organizados numa estrutura, especificamente, um grafo de conhecimento. Será utilizado um conjunto de dados obtidos no sítio de internet *Food.com*, que contém informações de receitas e classificações explícitas de itens, introduzidas pelo utilizador. A metodologia consiste na comparação de resultados entre algoritmos com técnicas híbridas e uma rede convolucional de grafos. A rede convolucional de grafos é a única que na nossa pesquisa, produz um modelo baseado num grafo de conhecimento construído à priori. Os resultados experimentais demonstram uma melhoria na precisão da previsão de classificação ao usar um grafo de conhecimento no sistema de recomendação.

Acknowledgements

Esta dissertação marca o ponto final do meu percurso no Mestrado Integrado em Engenharia Informática e Computação na FEUP. Mais do que marcar um ponto final, marca também a concretização de um pequeno sonho antigo, e como tal, estou profundamente grato a todas as pessoas que me ajudaram, apoiaram e me aconselharam.

Um agradecimento especial fica para a professora Ana Paula Rocha por ter aceite a orientação deste tema, pela ajuda, apoio e disponibilidade total em discutir o melhor caminho para a resolução dos vários desafios que foram surgindo durante a dissertação.

Agradeço à minha irmã e aos meus pais pelo seu apoio incondicional que me deram durante este percurso. Sem esse apoio não seria possível.

Ricardo G. Silva

'But I know, somehow,
that only when it is dark enough can you see the stars."

Martin Luther King Jr.

Contents

List of Figures

List of Tables

Abbreviations

ALS	Alternating Least Square
BPE	Byte-Pair Encoding
CNN	Convolution Neural Network
CSV	Comma Separated Values
DNN	Deep Neural Network
DBN	Deep Belief Network
GCN	Graph Convolutional Network
GNN	Graph Neural Network
KG	Knowledge Graph
MAE	Mean Absolute Error
MCCF	Multi-criteria Collaborative Filtering
MCDM	Multi-Criteria Decision Making
MCRS	Multi-Criteria Recommend System
MF	Matrix Factorization
NLP	Natural language Processing
RNN	Recurrent Neural Network
RS	Recommendation Systems
SVD	Singular Value decomposition
SVM	Support Vector Machine
ReLu	Rectified Linear Unit
RMSE	Root Mean Squared Error
TF-IDF	Term Frequency–Inverse Document Frequency
VSM	Vector Space Model
WWW	*World Wide Web*

Chapter 1

Introduction

The abundance of real-life applications with recommendation systems gives a clue to its importance in the digital user experience. The amount of information available and delivered to the user when looking for a product or service is overwhelming. Recommendation systems (RS) aim to provide a potential unknown item or service the user may desire for, to help users to understand their information needs better and efficiently to find items of interest or relevant to their needs. Methods and techniques evolved from the traditional recommendation systems like collaborative filtering based recommendation system, where utility function considers a single value, content-based filtering or hybrid filtering [29] to multi-criteria recommendation systems (MCRS). Many evolve from numeric statistics and similarity calculation to predication models.

MCRS is constructed on the principle of the incorporation of multiple criteria. When making a choice, a user can perceive the recommended item as suitable or not, depending on more than one utility-related aspect. Additional information improves the quality of recommendations in systems where recommendations are based on the opinion of others and where the incorporation of multiple criteria can lead to more accurate recommendations. Typical multi-criteria information comes from an explicit data survey where a user gives a rating to each aspect of the utility-related when it is related to services, or from provided ratings on multiple attributes when it is related to items. [5] For example, a movie recommendation system could allow users to rate movie attributes: plot, visual settings, music set. If every attribute has the same numerical interval, then it is possible to have a variety of rating combinations for the same movie. Other multi-criteria information can be the ratings reproduced by the performance of sentiment analysis tasks and machine translation tasks in user reviews when commenting on a recipe.

The importance of multi-criteria information reflects another attempt to catch more meaningful patterns to help predicting a user preference. The use of knowledge graphs for integrated data in a structured way provides logical inference of implicit knowledge and thus helping to identify new patterns. Research in recommendation systems already started making steps to apply these graph structure into the predicting models.

This paper presents an implementation of a recipe recommendation systems each analyses all information structured in a knowledge graph to build o model that can predict a rating given by

the user for an unrated recipe. The present approach focuses on giving a comparison between techniques widely used in collaborative filtering methods and a knowledge graph approach with convolution networks when dealing with explicit rating on a dataset whose items are recipes. Most of the research done in the recommendation systems field exposes techniques and construct models based on four main datasets where items are movies, music, books or people. The main reason is because the research lab in the Department of Computer Science and Engineering at the University of Minnesota, also known as *GroupLens*, updates and makes available the respective datasets for research purposes. The advantage is the quality of data present in the datasets.

Another reason to explore the application of the knowledge graph and graph convolutional networks is the lack of experiments in food domain. Considering recipes recommendation systems, most researches improve their models by introducing an improvement in calculation of the distance or by adding a new information dimensionality to the model, that is adding new attributes to the dataset. The improvements in distance measures focus on similarity in both sides, users side and item side. The improvements in models focusing on adding information goes from text, image or nutritional values to adding sources of information.

This research hopes to unveil a new step in the knowledge that can lead to a better recipe recommendation system. This dissertation includes five chapters, which are summarised as follows. Chapter 2 analyses the literature and presents a review of current recommend systems for recipes, other recommend solutions which take full potential from multi-attribute and multi-rating and solutions that employ knowledge graphs in neural networks and recommendation systems. Chapter 3 exposes the description, and analysis the problem domain for a recipe recommendation system. Chapter 4 describes the steps taken to implement a knowledge graph-based recipe recommendation system and metrics used. Chapter 5 display the results and specify the conditions and parameters of this experience, and, lastly, Chapter 6 finishes with respective conclusions of the research developed in this dissertation and possible directions for future work.

Chapter 2

Literature Review

As soon as recommendation systems evolved from e-commerce to services providers, food recommendation systems start to be a topic of research to change the eating behaviour, in many cases to aim for healthier food choices and to fight obesity. Various research methods and techniques focus on solving common issues on recommendation systems and improving the accuracy of those systems. This chapter addresses the fundamental concepts behind a recommendation system, exposes the present research in information filtering systems, and analyses the latest research in recipes recommendation systems. Section 2.1 describes the current structure in recommendation systems, giving an insight into the main techniques such as collaborative filtering and content filtering. Section 2.2 focuses on research with an approach in multi-rating and multi-criteria settings. Section 2.3 exposes related works using machine learning models and knowledge graphs. Finally, section **??** narrows the literature review into food domain recommendation systems.

2.1 General Approaches on Recommendation Systems

Recommendation systems are information filtering systems that aim to give a solution for the problem of information overload [33] by filtering vital information based on the user's preferences, interests, observed behaviour about an item or recorded behaviour on a web platform. [47] [29]. The types of information filtering systems split into three main groups: content-based filtering, collaborative filtering and hybrid, a fusion of the previous two. Over the last years, the innovation of such systems leads to a new categorisation based on techniques used: memory-based and model-based. The first category relies on similarity measures to match similar users or items together. In contrast, the second employs machine learning algorithms to train on the set of items for a specific user, and then build a predicting model whose objective is to predict the rating given by the user for a new item.

A tree view of the general structure of recommendation systems techniques is presented in Figure 2.1.

The next sections describe the two main groups, namely, content-based filtering and collaborative filtering; and reveal the main approaches and techniques used across the literature. As

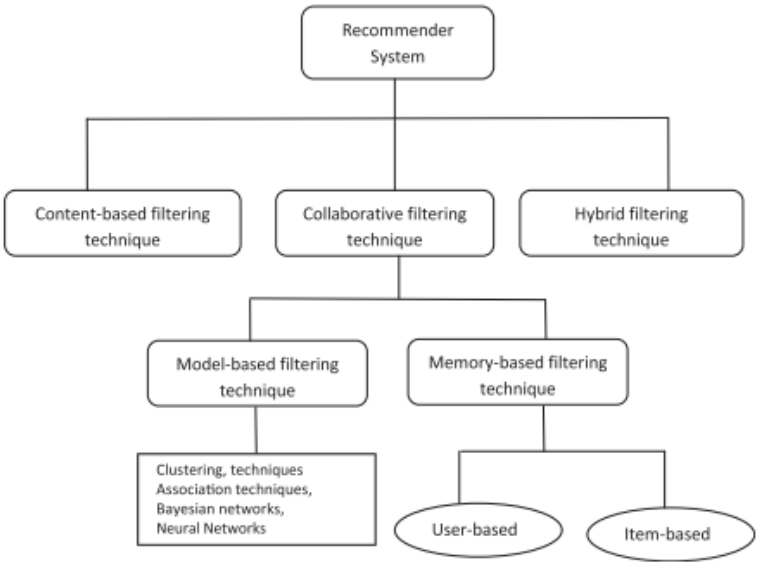

Figure 2.1: Tree view of Recommendation Systems techniques

information filtering systems try to evolve into more accurate systems, there are some particularities to consider, such as multi-criteria information systems, described in the Section 2.2. Section 2.3 describes in more detail the model-based techniques.

2.1.1 Content-Based Filtering

Content-based filtering makes a recommendation based on the similarity of items present in the user's history. The set of items in a user's history represents his interests. The system will look for items with the same attributes or properties of the object in the profile.

To build such systems, three steps are followed: *content analyser* where information extraction and data mining is performed, *profile learner* to construct a user profile and *filtering component* to compare between users profiles and candidate items.

The models most used to shape the user's profile are Naive Bayes Classifier, Rocchio Algorithm, Decision Trees and Nearest Neighbour algorithms.

After the profile construction, the filtering mechanism will compute similarities between the users and the items. The set of items recommended follow two rules: first, the highest similarity with a given user, and second, the highest similarity with the other items the user liked, viewed or rated.

The similarity calculation can have two distinct approaches: similarity-based metrics and distance-based metrics. The most common metrics in similarity-based are: the Jaccard similarity, useful for cases where the vectors contain binary values; the Cosine similarity, a calculus based on the cosine between the user vector and the item vector, also called user-item similarity.; and the Pearson's correlation which measures the linear correlation between a pair of quantitative and

continuous variables. On the distance-based metrics the most used are the Euclidean Distance and Manhattan Distance.

The limitations of these systems are the limit number of features that can be associated with one item, over specialisation or inability to find an unexpected item, designated as serendipity problem; and system inability to provide a recommendation for new users, known as a cold-start problem.

Most research in content-based filtering systems tries to improve the features of an item by adding more information. Main strategies used are the Vector Space Model (VSM) with basic TF-IDF weighting, keyword-based, semantic analysis with ontology and semantic analysis by using encyclopedic knowledge sources [38].

The general view of the process architecture of a content-based filtering system is exposed in Figure 2.2.

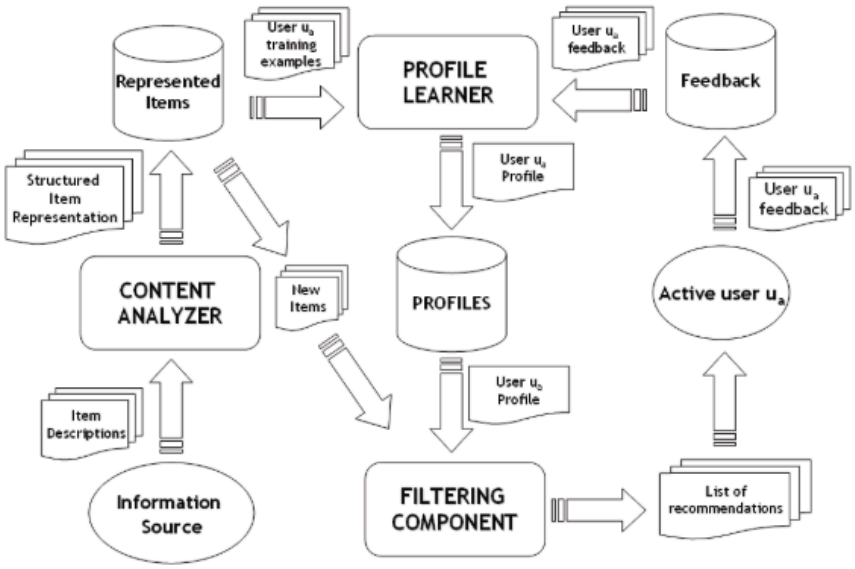

Figure 2.2: Architecture of a content-filtering system [38]

2.1.2 Collaborative Filtering

Collaborative filtering works with the user's item preferences and interests to build clusters of similar users. The preeminent concept is the neighbourhood. A rating translates a user preference in every position of a user-item matrix, as shown in Figure 2.3. The recommended item will be one unrated or undiscovered and one that the neighbourhood rated positively. [29]

Collaborative filtering evolved side by side with content-based filtering, joining in both memory-based technique and model-based technique in most cases.

Finally, the need to attenuate the disadvantages and enhance the advantages in both types of filtering led to the hybrid models. The state of work in hybrid recommendation systems shows the

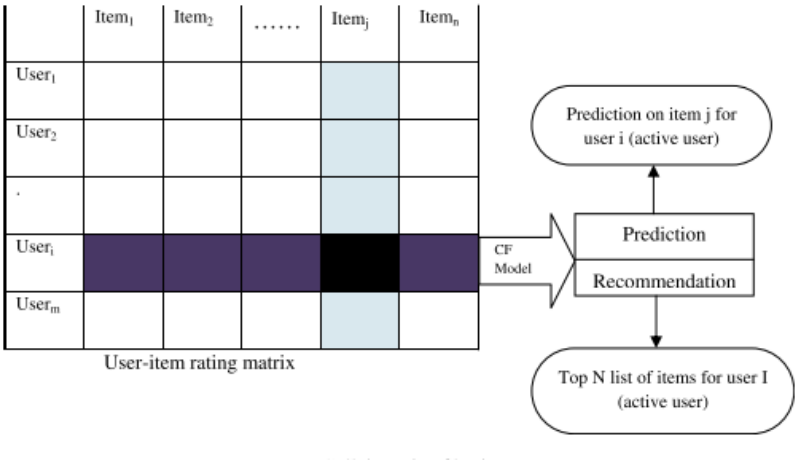

Figure 2.3: Collaborative filtering process [29]

research evolves around four main issues: cold start problem, sparsity, accuracy and scalability. New techniques evolve around the hybrid methods, a composition of collaborative and content filtering in both sides, user side and item side. This introduction helps understanding the approaches taken in the next chapters.

2.2 Multi-criteria and Multi-rating Recommendations

Another branch of development focuses on the desire to add contextual information or analyse different qualities or criteria on the items or the users. The idea is to increase the available information, quality and ideally limit the usual issues in recommended systems. [11] Two approaches, covered in the next sections are gaining popularity: multi-criteria recommendation systems and the use of knowledge graphs in model-based implementations.

User's information is available more than ever, and it gave the possibility to incorporate more information about the user and multiple views or dimensions of information about an item. A multi-criteria recommendation system tries a combination of these dimensions of information from multiple sources. The merging of these dimensions change depending on the type of item and its characteristics, thus enabling diverse multi-criteria implementations.

The construction of multiple criteria depends and changes significantly according to the type of item, be it a service or a physical item. For example, when buying a product, a customer can give more importance to the final cost or brand associated, instead of the delivery time. This changes significantly when considering a food delivery service, where some customers can give high importance to the delivery time, while other customers can have a higher preference for the quality of the food. Adding more contextual information about the user's preferences and implicit information about the user's behaviour leads to good results in multi-criteria recommendation systems. Multi-criteria recommendation provide strategies to deal with two vulnerabilities in recommend systems: the cold-start and sparsity problems. [46] [39] [22] [23]

Multi-rating recommendation systems are a way to modulate the user's opinion in service's sub components. The ratings of the service's sub components are explicit information, given by the user at the moment he gave the rating to the service. The most well known is *TripAdvisor* web service [31], where a restaurant is rated separately in three sub components: food, service and value, and the final rating is computed as shown in Figure 2.4.

Multi-criteria recommendation systems works similar to the following assumption, illustrated through a recipe recommendation. Suppose the existence of five users $u_1, \ldots, u_5$ and five recipes $i_1, \ldots, i_5$, and an unknown rating $R(u_1, i_5)$. $R(u_1, i_5)$ is the rating to predict. A recipe can be evaluated according to four criteria: meal preparation, ingredients, difficulty level and level of nutritional meal. Figure 2.5 also shows the rating given by the users to different criteria of the recipes. For the recipe, i_1, user u_5 gave a rating of `3` to meal preparation, a rating of `3` for ingredients, a rating of `9` for difficulty level and level of the nutritional meal. In this case, the overall rating of u_5 to the item i_1 is the average of the four criteria ratings previously noted, but it is possible to have different metrics like the cosine similarity, the Euclidean distance or even the Manhattan distance. In a traditional recipe recommendation system the prediction will find users that rated i_5 as well as similar users of u_1. What the multi-criteria information shows in this case is that user u_2 and user u_3 are completely different from u_1. For the recipe i_1, users u_2 and u_3 give a low rate to difficulty level and nutritional meal level. Multi-criteria rating concludes that u_4 and u_5 have more similarity for user i_1, and thus the RS will predict for $R(u1, i5)$ a rating of `5`. Multi-criteria have

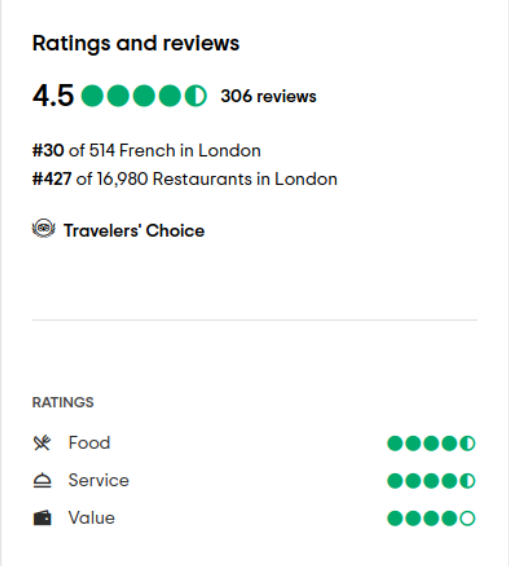

Figure 2.4: Example of an explicit multi-rating service: *TripAdvisor*

the possibility to give insights about the reasons why a user give a determined rating. [3]

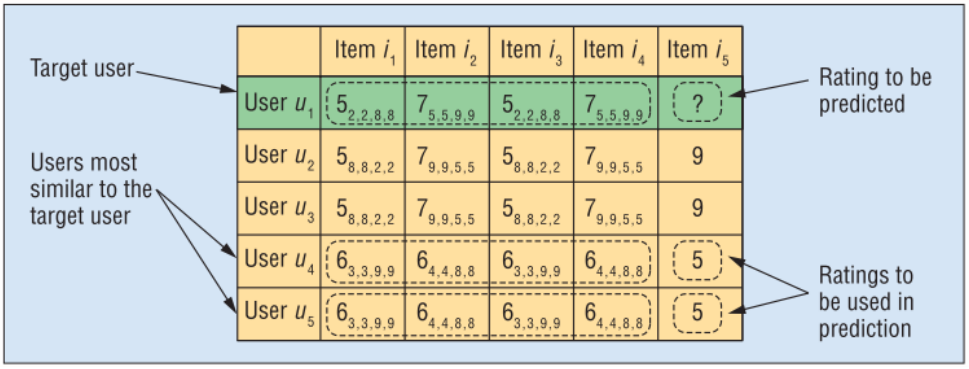

Figure 2.5: Matrix *User X Item* [3]

Manouselis et al. [4] and Adomavicius et al. [3] make a correlation between multi-criteria recommendation system as a multi-criteria decision making problem to define the category of multi-criteria rating recommendation as a set of techniques which will model a user's utility for an item in a vector of ratings following several criteria. A few years before, they also had identified the immediate necessity for complex modelling techniques as combinatorial and multi-objective optimisation ones [67] to be used by multi-criteria recommendation systems.[41]

Furthermore, they had concluded that most multi-criteria recommendation systems produces a list of recommended items to the users without considering other recommendation tasks.

Admavicius et al. [3] classified multi-criteria Collaborative Filtering (MCCF) based RS into two types: *memory-based* and *model-based*, also called aggregation function-based.

The first technique, memory-based, uses heuristics that calculate recommendations from previous user activities. This technique shapes the recommendation problem as an optimisation problem. For this, the general working steps are:

1. Calculate Pareto efficient solutions;
2. Making multiple criteria combinations and transform into single-criterion problem;
3. Optimising the most critical criterion and transform all the other criteria to constraints;
4. Optimising one criterion at each time;
5. Take optimal solution and change it into constraint;
6. Repeating for the remain criteria.

The second technique, model-based, builds a predictive model from user activities and, after, uses this model to make recommendations. The predictive model is built using machine learning techniques, and the most commonly used are clustering, K-nearest neighbour, neural network, fuzzy logic, evolutionary computing and matrix factorisation. [25]

Adomavicius [4] provided an overview and a depth analysis of the majority multi-criteria recommendation systems, which he classifies in three general categories.

- **Multi-attribute content preference modelling**: For each user, the recommend system studies the items preferred in the past by the user and then it will try to model and understand similarities between any given of shared set of multi-attribute. The recommendation output will be a list of best matching items.
- **Multi-attribute content search and filtering**: The system behaves like the previous one with two complements: searching or filtering options. The system uses the information inserted on search or filtering and reduces the set by finding similarities in content-based attributes in all items. The recommendation output will be a list of best matching items and items that satisfied the search or filtering conditions.
- **Multi-criteria rating-based preference elicitation**: The system makes it possible to define individuals preferences. For each item, a user gives individual rating in multiple criteria. The recommendation output will be a list of best matching items that are in accordance with the preferences but as well with the overall multi-criteria rating from other users.

2.3 Model-based Systems with Graphs Structures

Many model-based systems rely on deep learning architectures. Deep learning has become a strong presence in recommendations systems. Before describing existing research in deep learning architectures with knowledge graph structures, it is essential to remember the fundamental

concepts of deep learning (Section 2.3.1) and knowledge graphs (Section 2.3.2). Then, Section 2.4 presents existing recommendation systems research in the field of recipes that uses knowledge graph structures and neural networks.

2.3.1 Deep Learning Architectures

Deep learning has one important concept: perceptron. A perceptron, or a single neuron, is a module which operates in the following way: a set of inputs, also called features, are multiplied by the corresponding weights, summed up, and the result is applied into a generally non-linear activation function, producing an output result $\hat{y}$, as shown on Figure 2.6. A non-linear activation function is a function that takes any real number as input on the x-axis and it transforms that real number into a scalar output between zero and one. Neural networks has many types of activation functions, being the most known: *Sigmoid*, *rectified linear unit (ReLu)* and *softmax*.

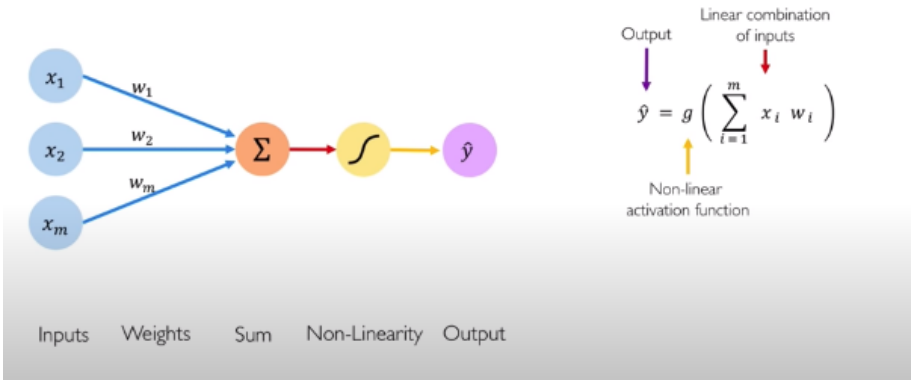

Figure 2.6: Perceptron composition

An equally important concept is the bias term. Its purpose is to shift the activation function in opposite directions regardless of the inputs in the neuron. The inputs and the weights are vectors and the activation function will compute the dot product of the vectors plus the bias term, resulting in an output $\hat{y}$.

Another concept to discuss is the layer. Multiple inputs connecting to multiple perceptrons, results in multiple outputs, and that creates a layer. The set of weights of each connection to an input is different. When all inputs are densely connected to all of the perceptrons these layers are called dense layers. Figure 2.7 shows a single layer neural network.

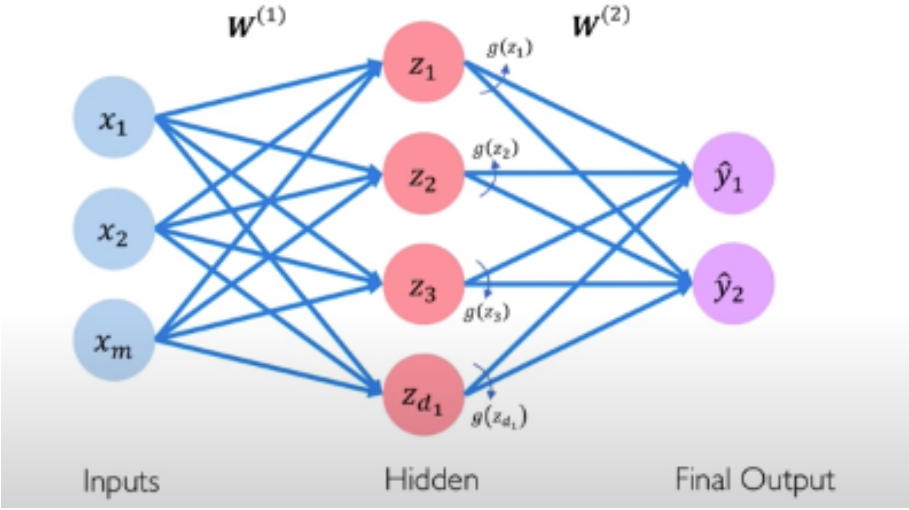

Figure 2.7: Single Layer Neural Network

When defining a neural network to be applied to a specific problem, the training of the model is a crucial task, it's where learning takes place. The training is performed on a historical dataset. The training will discover the weights, or the vector of weights, that try to minimise the total loss over the entire dataset and then validate our model. The optimisation of the neural networks is possible through changing the learning rate, i.e, changing the gradient descent and mini-batches. The gradient descent is another important concept. It will analyse how the loss performs with respect to each of the weights in order to understand the direction to a local minimum loss. Mini-batches solve the high complexity computation time in doing optimisation in just a small portion of points to understand how loss performs.

The concepts introduced are shared along with distinct types of deep learning architectures. Deng [12] identified five major types: deep neural networks (DNN), deep belief networks (DPN), recurrent neural networks (RNN), convolution neural networks (CNN) and graph neural network (GNN). Concerning this research's objective, only convolution neural network (CNN) and graph neural network (GNN) are analysed because their ideas led to the birth of graph convolutional networks (GCN) and, later, the use of knowledge graphs in neural networks.

The Convolution Neural Network was born as a specialised branch of neural networks to deal with image and video processing. It used the vector of pixel values as an input to learn image features. By connecting small image patches of the input to a single neuron in the hidden layer, it is possible to maintain the spatial structure. Convolution neural network use a sliding patch window, also known as filter, across the input image and across the subsequent layers to define connection between patches and, thus, extract features and learn weights. This operation is called convolution. The general idea can be summarised in the following points: learning features in input image through convolution; introduce non-linearity through activation function; and, reduce dimensionality and preserve spatial invariance with pooling operations.

The Graph Neural Network is a neural network specific to operate in a graph structure which can be either cyclic, directed, undirected or a mixture of these. Essentially, every node in the graph is associated with a label and the objective is to predict the label of the nodes without ground-truth. [50] The downside of this architecture is the quadratic complexity as stated in Atwood et al. [6]

These two types of neural networks inspired Kipf et al. [32] to introduce the concept of Graph Convolutional Network (GCN) and propose an efficient variant of convolutional neural networks which operate directly on the graph.

The Graph Convolutional Network takes the same abstract idea from convolution neural networks but applies it into graph structures data. In other words, for each node, the feature information from all its neighbours and itself is fetched and sent as input into the neural network. Figure 2.8 shows a graphic view of the convolutional idea behind convolution neural network and graph convolutional neural network.

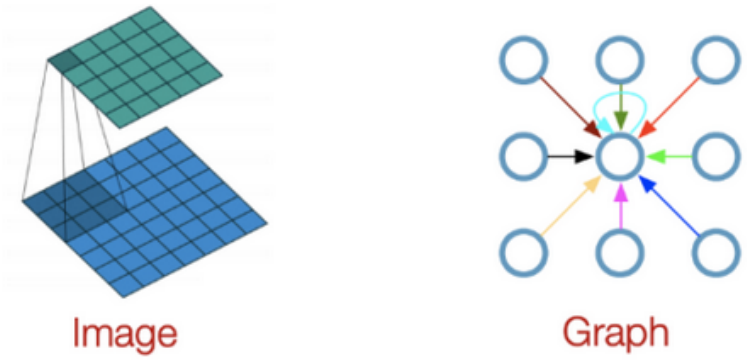

Figure 2.8: Convolution idea: Single CNN layer with a 3x3 filter and node neighbours aggregation

Particularly, a GCN takes graph data and classifies the vertices according to some node label. It will take an input as a graph and sequentially process it through the hidden layers and flatten these representations out into a *softmax* activation function over labels.

The number of layers is the farthest distance that node features can travel. For example, with one layer GCN, each node can only get the information from its neighbours, and can not get information from the neighbours' neighbours. The gathering information process takes place independently at the same time for all the nodes.

When stacking another layer on top of the first one, the algorithm repeats the gathering information process, but this time the neighbours already have information about their own neighbours (from the previous step). It makes the number of layers as the maximum number of hops that each node can travel. Usually, a configuration with six or seven hops, will get almost the entire graph which makes the aggregation less meaningful.

Kipf et al. [32] introduces the corresponding propagation rule between layers in Equation 2.1

$$H^{\updownarrow+1} = \sigma(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(\updownarrow)}W^{(\updownarrow)}) \tag{2.1}$$

The hidden features at the next layer, H^{l+1}, are defined as being a non linear activation, σ, by the term $\tilde{A}\tilde{D}$ multiplied by the previous layer activations, H^l, and by the weight matrix for this layer, W^l. The term $\tilde{A}$ is the adjacency matrix where an adjacency matrix is how you define the edges in the graph plus added self loops. The self loops in nodes are added for two reasons: so that each node includes its own features at the next representations and to help with numerical stability. $\tilde{D}$ is the degree matrix of $\tilde{A}$ with a function of normalised nodes with large degrees because otherwise nodes that have a high number of neighbours would have a corresponding high magnitude in the features. The reason behind this behaviour is because every time it propagates forward, features are being aggregated with many other nodes. Other key idea presented by Kipf et al. [32] is symmetric normalisation, i.e, it uses the half negative of $\hat{D}$ and puts it on each side of $\hat{A}$.

Others works worth of mention are the research of Duvenaud et al. [14], Li et al. [37], and Jain et al. [30] which introduced specialised architectures that adapt RNN and CNN to work on arbitrarily structured graphs. Alternatively, Bruna et al. [10] and Henaff et al. [26] went to spectral graph theory and used graph convolutions to define parameterised filters in CNN, and thus applied CNN to structured graphs.

2.3.2 Knowledge Graph in Recommendation Systems

The term Knowledge Graph (KG) widespread when Google, in 2012, announced the use of semantic knowledge in web search with the famous statement *"Things, not strings"*. It is an approach to achieve a semantic integration from heterogeneous data sources. A Knowledge Graph allows the creation of structures to categorise and tag the content properly. The knowledge graph structures itself in relationships according to semantic standards, and can easily and directly link the knowledge graph relationship to everyday language. Two main direct advantages are a high organisation of information and descriptive information that helps infer new knowledge from the graph.

A knowledge graph is a heterogeneous graph-based representation of knowledge composed by entities, the nodes, and relations between entities, the edges. The format of an edge is a triple composed by head entity, relation and tail entity. The triple is called a fact and represents the relationship between the head entity and tail entity. From Figure 2.9, that presents an example of a knowledge graph for music, a triple can be <Shape of you, written by, Ed Sheeran> and it shows the relationship "song *Shape of you* is written by *Ed Sheeran*". The set of possible types and relations define an ontology, and it gives an understanding of the relationship and restrictions between entities. [48]

As recommendation systems try to use more information, knowledge graphs appear a natural path of research. In fact, in a knowledge graph, it is possible to map items, it's attributes, users, preferences and relations between users and items. Preliminary research shows that the use of knowledge graphs give more accurate recommendations and facilitates the interpretability of results. [24]

Figure 2.9 helps to understand the concept behind interpretability of recommendations. The figure shows a set of different entities: title of music, genre, person. And each of these entities is

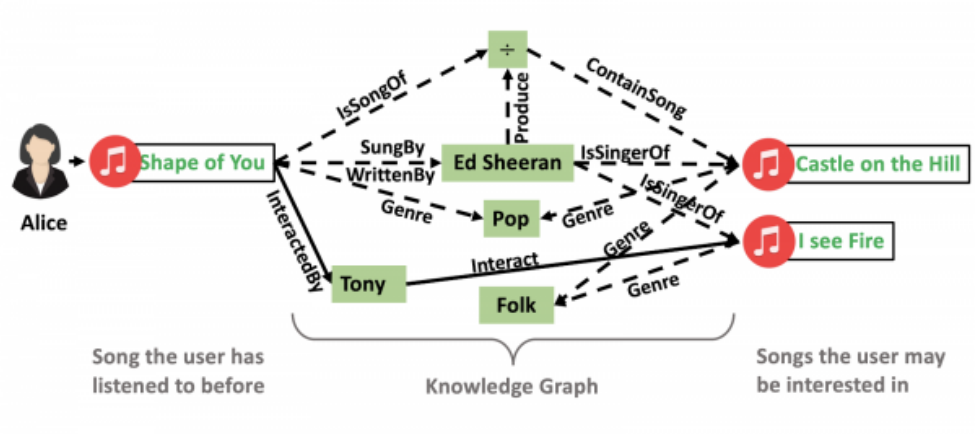

Figure 2.9: Example of a knowledge graph for music

instantiated. For example, title of music entity has the value *Shape of You*, *Castle on the Hill* and *I see Fire*. Entity Person has two values: *Ed Sheeran* and *Tony*. For the edges of the knowledge graph, the following relations between entities can be found: *IsSongOf*, *WrittenBy*, *Genre*, *Interact*, *IsSingerOf* and so on. Alice's recommendation is based on the common information between *Shape of You* and *Castle on the Hill*. Both share the same song writer, genre and album. The second Alice's recommendation, *I see fire*, will be based on the connection *Tony*, the same song writer, and the *Folk*genre.

Knowledge graphs in recommendation systems rely heavily on embeddings. Embedding is a method to encode textual information into a vector. A graph embedding determines a fixed length vector representation for each entity (usually nodes) in our graph. These embeddings are a lower dimensional representation of the graph and preserve the graph's topology. It can be seen as a tool to create structure around the low dimensional real vectors in a way that similar items are nearby, and then the structure is in fact geared towards the objective of the recommendation system. Embedding can also be applied to dense data (e.g audio) to create meaningful similarity metric and, furthermore, it can embed diverse types of data (e.g. texts, images, audios) jointly and learn similarity metric across them. Embedding is a key concept to understand the recent strategies around how knowledge graphs can leverage recommendation systems.

Guo et al. [24] identify three types of knowledge graph recommendation systems: the embedding-based method, the path-based method and the unified method.

- **Embedding-based method**: the knowledge graph have a rich representation of items, through adding different knowledge sources. For example, Zhang et al. [68] embedded textual knowledge (extract from comments) and visual knowledge (extract from images) besides the usual structural knowledge (characteristics of the item) into the whole knowledge graph. Another approach is embedding the user preferences in the graph and therefore build an user-item graph. The triples are the representation of users behaviours and item

properties, as shown in Figure 2.10 [69].

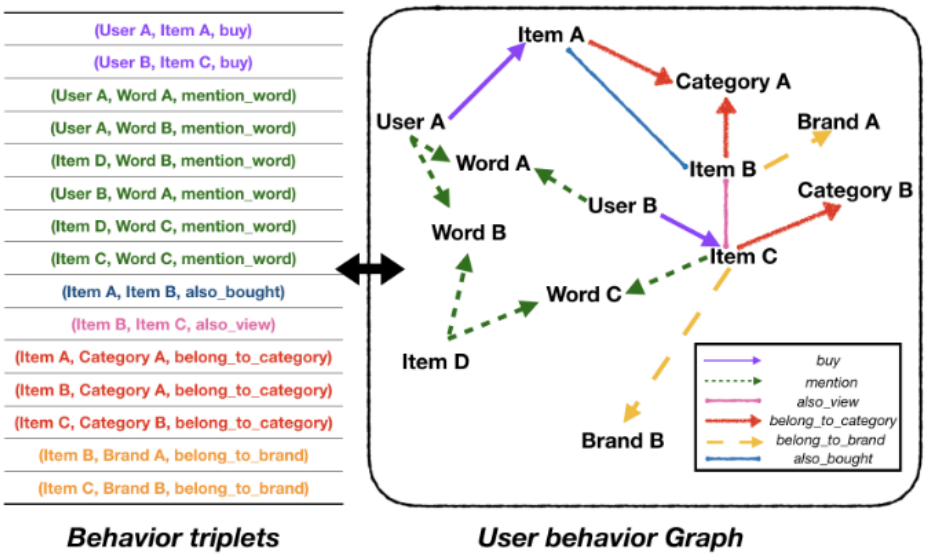

Figure 2.10: User-item embedding knowledge graph demonstration [69]

- **Path-based method**: build a user-item graph as in the embedding based method, but the recommendations are based on the connectivity similarity of users and/or times to enhance the recommendations [24]. The basic idea is that two entities can be connected via different paths. For example, two actors can be connected via "actor-movie-actor" path, "actor-movie-category-movie-actor" path, and so on. As stated by Sun et al. [54] "the semantics underneath different paths imply different similarities". In another words, an entity representation will be enriched by the path connectivity.

- **Unified method**: the unified method is the combination of the embedding-based method and the path-based method.

Guo et al. [24] in their survey give a comprehensive overview over the knowledge graph-based recommendation system showing that researchers prefer conjunction of algorithms in pairs between convolutional neural networks, recurrent neural networks, attention mechanism and graph neural network, and matrix factorisation. Table 2.1 selects the most valuable approaches given the objective of this research.

Note that Attention Mechanism and Graph Neural Network are used among all the unified knowledge graph recommendation systems. The Attention Mechanism emerged as an improvement over the encoder decoder-based neural machine translation system in natural language processing (NLP). [7]

Approach	**Functionality**
KGCN-LS (GCN+ Attention Mechanism)	"Computes user-specific item embeddings by first applying a trainable function that identifies important knowledge graph relationships for a given user." [63]
KGAT (GNN+ Attention Mechanism)	"Heterogeneous network embedding method (TransR) to extract items'structural representations by considering the heterogeneity of both nodes and relationships. Stacked denoising auto-encoders and stacked convolutional auto-encoders were applied, which are two types of deep learning based embedding techniques, to extract items textual representations and visual representations, respectively." [66]
AKUPM (Attention Mechanism)	" Investigates how to explore relationships between entities and users which are essentially determined by the interactions among entities." [56]
KNI (GNN + Attention Mechanism)	"Propose neighbourhood interaction model to capture each neighbour pair (between user-side and item-side) distinctively." [49]
RCoLM (Attention Mechanism)	"Takes the incompleteness nature of KGs into consideration when incorporating them into recommendation. Attempts to exploit user-item interactions from recommendations for KG completion, and unifies the two tasks in a joint model for mutual enhancements. Secondly, RCoLM provides a general task-oriented negative sampling strategy on KG completion task." [35]
AKGE (GNN + Attention Mechanism)	"Automatically extracts high-order subgraphs that link user-item pairs with rich semantics, and then encodes the subgraphs by the proposed attentive graph neural network to learn accurate user preference." [51]
IntentGC (GNN)	"Leverage both explicit preferences and heterogeneous relationships by graph convolutional networks." [70]

Table 2.1: Deep learning approaches in knowledge graphs

2.4 Recipes Recommendation Systems

Recommendation of recipes evolve with the increase techniques in domains like online shopping, audio and video content, and even social networks by suggesting friends. Research in recipes recommendation has increased adopting a hybrid approach. Tran and Ants [59] continue the work of Mika [42] to summarise the main existence recommendation systems related to food, in four types, when considering systems which provide healthy food suggestions: collaborative approach using user preferences, collaborative approach with external information, hybrid approach and group recommendation.

The first type uses a collaborative approach and focuses on user preferences. The goal is a recommendation of healthier recipes or food items, much similar to the ones the user liked in the past. These collaborative filtering systems work with information from a set of user clusters, food categories and ingredients.

El-Dosuky [15] applied a TF-IDF algorithm (term extraction method with cosine similarity measure) to identify similarities between a user profile and a recipe. Two critical factors can be pointed out in this approach: first, the incorporation of a standard food database and second, the need for the user to give a rating in food items of different categories.

Freyne and Berkovsky [19] use the average of all the rating values of all ingredients included in the recipe to predict a rating value for a target recipe, and conclude it is possible to achieve better results with a sparse matrix when considering an item as a set of ingredients.

Elahi [16] presents a different perspective in user preferences. It considers essential to distinguish between long-term and short term user preferences to later applied on Matrix Factorization (MF) rating prediction model. The framework consists on tagging and rating recipes most useful to the recommendation system with an Active Learning Algorithm (long-term preferences) to connect with the short-term preferences(session-based information like a list of ingredients the user wants to cook).

Kuo et al. [34] transform a menu planning problem into an optimisation problem making use of an undirected recipe graph, being each node has a set of ingredients, the recipe distance is the similarity between two recipes and cost. The Connector-Steiner Tree Algorithm generates a subset of recipes to construct the menu.

The second type uses the same collaborative approach of the first type but adds a new layer of information, for example, information from a health care provider, or nutritional needs for health problems [2]. Ueta et al. [60] provided an extensive overview of what can be achieved if more data is added, such as dietary restrictions and nutritional values.

The third type is a combination of the two previous types [53], and the fourth type specifically deals with group recommendations rather than individuals [9].

Lastly, another research not fitting in the above categories but important to mention is a recommendation of the most similar recipe as in [58], where authors use networks for ingredients to distinguish them between complementary ingredients and substitute ingredients. With the use of clustering algorithms, is calculated the probability of two ingredients appearing simultaneously in

the same recipe against the probability of occurring separately. The learning methods used in this work were the support vector machine (SVM) and stochastic gradient boosting trees.

Nevertheless, research is still going on arising new challenges in collecting user information and improving the overall model, increasing information of recipes with nutrition data, recommendation algorithms, explanatory recommendations for each result, generating recommendations with group decision information and recommendation systems whose goal is modifying eating behaviours.

For the objective of this research, the study of the last challenges solved is crucial because it presents us with some approaches or perspectives that can increase the quality of the solution of a recipe recommendation systems. Table 2.2 summarises the approaches and goals found in existing recommendation systems in the culinary domain. Collaborative Filtering approach is the most used in recommendation system and it is one of the choices when the intention is joining two different approaches. Table 2.3 presents the solution adopted in existing recommendation systems to a list of the most relevant challenges in food consumption.

Recommendation approaches	**Functionality**
Knowledge-based recommendations	"Proposing a framework for a personalised nutrition service." [15]
CF, CB, hybrid recommendation	"Predicting item ratings by breaking down recipes into ingredients and vice-versa." [19]
CF,CB,hybrid recommendation	"Improving the quality of recommendations by using machine learning techniques and an understanding of user reasoning." [20]
CF	"Developing an online grocery store to provide users with recipe recommendation by analysing the social navigation in groups." [55]
Matrix factorisation	"ChefPad - Generating food recommendations by eliciting users' long-term and short-term preferences." [16]
Graph-based recommendation	"Recommending sets of recipes by using user-specified ingredients." [34]
Goal-oriented recipe recommendation which suggests the right type of nutrient to treat users' health problems Hybrid	"Helping users to deal with health problems." [60]
Hybrid recommendation (CB CF), Constraint-based recommendation CB	"Meal Planning System - Aiding the elderly to deal with malnutrition problems and change the food consumption behaviour." [2]
CF, group-based recommendation	"Applying different aggregation strategies and user weighting models to generate recipe recommendations to a group of users." [9]
Group recommendation, Critique based conversational recommendation	"Generating food recommendations to groups by exploiting users tags and ratings." [17]

Table 2.2: List of approaches and functionalities in food RS

Research challenges	**Proposed Solutions**
Collecting user information	– "Taking advantage of information about users' previous meals." [61] – "Implicitly collecting user information so that they don't have to invest too much time and effort." [19]
Gathering nutritional information of recipes	"The quantity of gathered recipes should be representative enough to vary the recommendations." [60] [2]
Recommendation algorithms	"Improving the quality of recommendations by integrating constraints (e.g., health situations, nutrition needs, the availability of ingredients) into the recommendation process." [15] [59]
Explaining recommendations	"Providing explanations which increase the trustworthiness of decision outcomes and persuade users to accept food recommendations." [9]
Generating bundle recommendations	"Expressing acceptable trade-offs among group members by employing negotiation and argumentation mechanisms." [18]
Achieving fast consensus in group decision making	"Enriching user interfaces supporting basic negotiation mechanisms among group members." [17]
Changing eating behaviours	– "Employing health psychology theory in food recommendation systems to encourage users to comply healthy eating behaviours." [52] – "Proposing potential dietary changes on the basis of exploiting the ideal nutrients from reliable resources (e.g., USDA, DACH)." [34]

Table 2.3: Last challenges and solutions proposed in food consumption field

2.5 Summary

As the e-commerce and web services rise in the amount of content available and the number of users, recommendation systems became an important research topic. They perform a crucial rule in user experience and content discovery. So crucial that is difficult to find a web service without some recommendation system. User behaviour gives vital importance to personalisation and does not want to lose time in research and find new content for his needs. For example, online social media and social networking service are a fundamental business with various recommendation systems as a part of his ecosystem. The amount of data available and the increase in hardware performance lead to a shift in recommendation systems. Instead of memory-based systems with the content or collaborative filtering, model-based systems became more predominant alongside with machine learning developments. Model-based systems became very powerful and accurate as more item's data and user's data is added, or even new measures and approximations techniques have been deployed. Knowledge graph-based systems are pushing forward new developments in recommendation systems in their specific way of information representation, and preliminary results forecast improvements in recommendation systems field.

As stated before, an important factor in using knowledge graphs is deep learning architecture. The architecture consists of multiple layers interconnected, each one with a specific stage and a hierarchical position that has the goal to accomplish pattern classification, feature selection, pattern and feature learning. Attentions Mechanism and Graph Neural Networks were the most common tools to make use of information structure in knowledge graphs in recommendation systems. Since then, graph convolutional network and variants have shown better metrics in various application areas on distinct datasets.

Chapter 3

Problem Description

This chapter specifies the problem characteristics given the dataset obtain to accomplish the goal of this research. Section 3.1 defines the problem and puts in perspective potential advantages and disadvantages in knowledge graph. Section 3.2 presents the dataset characteristics and Section 3.3 gives an insight into the dataset used for this research.

3.1 Problem Description

A recommendation system helps users find compelling content in a specific domain that the user is interested in. It differs from the use of search engines, because a recommendation system can display items that users might not even have thought about searching on their own and are the high interest to them or have a high probability of being. A user-item interaction is a relationship between an user and an item, given by an explicit or an implicit feedback. A form of explicit feedback is a numeric rating scale and a form of implicit feedback can be, for example, a *like*, a textual review or a click in a specific link. An item is a final product or a service. The heterogeneity of the item will influence the recommendation system, because it has a context. For example, if it is a final product like books, calculating the similarity between items and clustering users may be enough, however, when it came to services like a hotel, the recommendation system may consider the cost, the degree of cleanliness or the reception service, as customers usually value these characteristics.

The preparation of a recipe comprises a great deal of information related to different features: cooking methods, cooking time, ingredients, quantities, difficulty and more. The problem addressed in this work is the development of a recommendation system that can help users identifying the most suitable recipe for their current preferences. Analogous to movies, the complexity of a recipe recommendation system increases when considering a recipe as a combination of ingredients, tags and steps. If considering vegetables used as ingredients alone, it is possible to count more than a thousand of vegetables [59]. The same happens when considering food recipes

tags or other complementary information that composes a recipe. Moreover, if we consider user preferences, it is vital to be aware of the complexity of the user's opinion, where the most undefined variable is the taste. User goals, cultural environment and social factors also have a strong influence in food choice. Multiple sources of information can be added to the prediction model inherent in the recommendation system.

The attempt to capture more information and considering item beyond its characteristics is a natural step. The focus of this research is precisely trying to capture recipes features and attributes in a structured way by using knowledge graphs.

Knowledge graphs are a tool for extract and deal with complementary information around an item. It can improve the overall recommendation system because it will take advantage of the semantic information and the high structure represented by the graph. Most research use knowledge graphs with unified methods with movies, books or music datasets to test their strategies. However, to the best of our knowledge, no prior attempt was done to model food recipes in knowledge graphs with unified methods.

Before translating a structure and semantic information into a knowledge graph, is essential to describe the problem in reference to consider an item as a set of combination of proprieties, attributes or features that can be analysed and considered for the model.

The knowledge graph is a directed graph $G = (V, E)$, where V is the set of nodes and E the set of edges. The nodes are the entities and the set V can have more than one entity type. Each edge on E is in the form of subject-property-object triple, $< e_h, r_{e_t} >$, also known as a triple of *(head entity, relation, tail entity)*. Each edge is interpreted in the following manner: entity e_H has a relationship of the type r_t with entity e_t. In knowledge graphs, a path defines one relation sequence, called meta-path, where $P = A_0 \xrightarrow{R_1} A_1 \xrightarrow{R_2} A_2 \xrightarrow{R_K} A_K$ which defines a set of relations $R_1 R_2 ... R_k$ between types A_0 and A_k where $R_i \in E$ and $A_i \in V$. The combination of meta-paths is called meta-graph. The meta-graphs will represent the neighbour's features for a particular node.

A knowledge graph for a recommendation system will need the user feedback to perform the link between users and items. To combine the user feedback and represent the user-item interaction, every interaction between user u_i and item v_j is translated into a binary relation type, where 1 is defined as an interaction observed, and 0 is defined as interaction not observed. The binary relation type can vary depending on the type of interaction being implicit or explicit. For explicit interaction that is based on a rating, it is common to define a limit on the rating scale, where lower rating scores are translated to a not observed interaction. The final knowledge graph with users, interactions and items information is characterised as an undirected heterogeneous graph.

The final step is to transform the knowledge graph and the previous binary relation with an embedding procedure. Every entity and relation will be transformed into a d-dimensional vector, and all the structure and high-order proximity in the graph will be preserved.

As Guo et al. [24] states, the general idea is the information from item's content and item's attributes, represent a latent vector $\mathbf{V}_j$ of each item v_j and the information from user-item interaction matrix represents a latent vector $\mathbf{U}_i$ of each user v_i. The probability of u_i selecting v_j is calculated

by the inner product between the two vectors, which is calculated in every neural network layer. Other alternatives for the neural network's function are measuring the distance between two entities or the latent vectors' sum.

The main advantage is the possibility to explore related hidden connections, to increase the result's precision, to extend diversity on the recommendation set and make the recommendation process more understandable. These advantages mitigate some issues related with traditional recommendation systems: the cold start problem and the sparsity [62]. The cold start problems happens when the system does not have enough information about an user or an item and, therefore, can not make predictions or recommend it. The sparsity concerns the values of the user-time matrix, where it is mainly filled with zeros, in other words, a very few interactions resulting in a sparse matrix.

Given the problem formulation, we desire to build a model which can predict a rating for a given user-item pair, and compare it against a test set. The result will be a a supervise machine learning with a prediction problem using a knowledge graph to structure all information. From the set of model techniques that can work with this specific structured information, three were considered: graph neural networks, graph convolution networks and knowledge graph based convolutional networks. Among the three, knowledge graph based convolutional networks proved to be the one with the best results considering the problem similarity, the explicit rating and this research's objective: build a recommendation system model [63].

3.2 Dataset Analysis

In the world's largest data science community, *Kaggle* [21], is possible to find recipes datasets that include not only information about recipes but also user information. From the recipes datasets available, was chosen the Food's dataset [36], originated from *Food* website.[57] This same dataset was used by Bodhisattwa et al. [40] in their research to generate personalised recipes. Another reason for the selection of this dataset is the similarity of its composition with the movies dataset [45] used on the information filtering research field.

Food's dataset contains recipe composition with tags and user reviews, users and items interactions and respective ratings as described on the following seven files in Comma Separated Values (CSV) format and respective attributes designation:

- `PP_recipes` - contains processed recipes information in the form of byte-pair encoding (BPE) via subword tokenizer (GPT): recipe identifier, contiguous recipe information, recipes name tokens, ingredients tokens, steps tokens, list of techniques used in recipe and identifier of ingredients in a recipe;

- `PP_users` - contains information about the ratings and reviews given by user: a user identifier, techniques in the recipes rated, list of recipes rated, the respective rating for each one of the recipes rated and the total number of rates;

- `RAW_interaction` - contains all information about a user-item interaction: a user identifier, recipe identifier, date of the rating, rating given by the user, contiguous user identifier and contiguous recipe identifier. This file is the source of the following three files also present in the dataset *interactions_test*, *interactions_train* and *interactions_validation*;

- `RAW_recipes` - contains all information associate with a given recipe: name, recipe identifier, minutes for preparation, contributor's identifier, date of the upload, descriptive tags, nutrition information, the total number of steps, preparation steps, recipe's ingredients and the total number of ingredients.

The *RAW_interaction* file is the base of three other files in accordance to the machine learning methodology steps: train, test, and validation. All *interactions* files contain the explicit rating attribute - an user gives to a recipe a score of 1 to 5, where 1 indicates an extreme dislike and 5 indicates an extreme like - but without the textual review attribute present in the original interactions file. The *raw_recipes* file contain all the information per recipe, while *PP_recipes* file is the recipe information in byte-pair encoding.

Inconsistent values between files can lead to ambiguity in perceiving and understanding objects and return inaccurate results. In order to improve data quality, check data consistency and integrity, the following steps were implemented:

- All files were dealt as *Dataframe* data structures and types were assigned as follows: identifiers were assigned as integers, textual entries were assigned as strings, and upload date and interaction date assigned as data-time format;

- In all files empty entries information were checked and in case of empty entries the entire row was deleted;

- In all files null or symbols entries information were checked and in case of null or symbols the entire row was deleted;

- All integers identifiers columns across all files were checked for duplicates and no duplicates were present in the dataset;

- The consistency between contiguous identifiers present in interaction files were checked with `PP_users.csv` and `PP_recipes.csv`. Identifiers not present in interaction prefix files were deleted;

- Rows duplication was checked across the three *interaction* prefix files because they are mutually exclusive. The duplication rule was based on the index composed by user id and recipe identifier. Duplicates rows were deleted.

Users and items identifiers have consistency among all files and on interactions files, the contiguous integer index for every user and item was maintained for later use in *Dataframe* data structures.

As a result, the final dataset characteristics are summarised as follows:

- 1132367 interactions between a user and a recipe in the form of explicit rating;
- 25076 users rated on this dataset;
- 178265 recipes available;
- For each entry, all attributes have values.

In next section, the data characteristics are exposed in more detail taking into account the information needed to accomplish the objective of this work, that is, the implementation of a recommendation system.

3.3 Data Exploratory

Detecting information in a dataset with unusual properties is essential as such information often means abnormal behaviour or pattern. The elimination or the reduction of these problems means a direct growth in the quality of information extracted by data analysis processes. To tackle this issue, outlier detection is an essential task to be performed. Interquartile range distance is a well known technique that identifies potential outliers.

Next sections describe in detail the analysis performed over the attributes found more relevant: preparation time, number of steps, number of ingredients, tag and rating and respective decisions concerning the outliers detected. Analysis of the interaction data is also performed.

Preparation Time

The attribute *minutes* is the recipe preparation time and it shows a small set of high values, 2859 samples (1,23% of the dataset) with more than 720 minutes. Further analysis revealed alcoholic and beverage recipes have the most time-consuming step which is fermentation. In other entries, the attribute minutes shows inconsistency with the description, as Irish Flummery recipe; whose time preparation takes three and half days by instructions and preparation is done in less than sixty minutes. Others corresponded to recipes where the recipe implies the preparation of ingredients overnight or is a slow-cooking method. To build a general model, the decision was to consider only recipes with a time preparation less or equal to twelve hours. Figure 3.1 shows the distribution of the food recipes' preparation time.

More than half of the recipes have one hour or less of preparation time. The number of recipes that requires between one hour and two hours is still relevant.

Number of steps

Another critical attribute is the number of steps because it provides clues to the recipe's difficulty or easiness. From another perspective, since this dataset has many recipes submitted by the users, many steps are synonymous with a worldly review by the user or a review on the recipe instead of the actual recipe descriptive steps. Only recipes with less or equal than 100 steps are maintained in

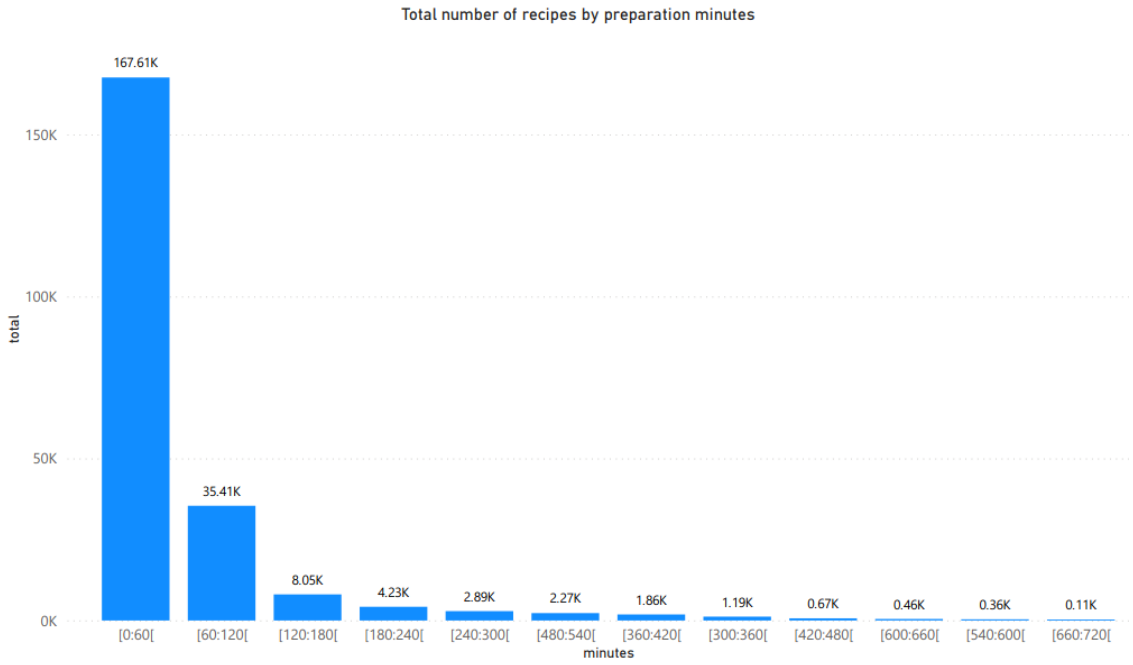

Figure 3.1: Total number of recipes by preparation time in minutes

the dataset. More than 100 steps were considered outliers, that translated in four recipes, 0,001% of the dataset. Figure 3.2 shows that most of the recipes can be accomplished with less than ten steps, and close to one quarter of the total requires between ten and twenty steps.

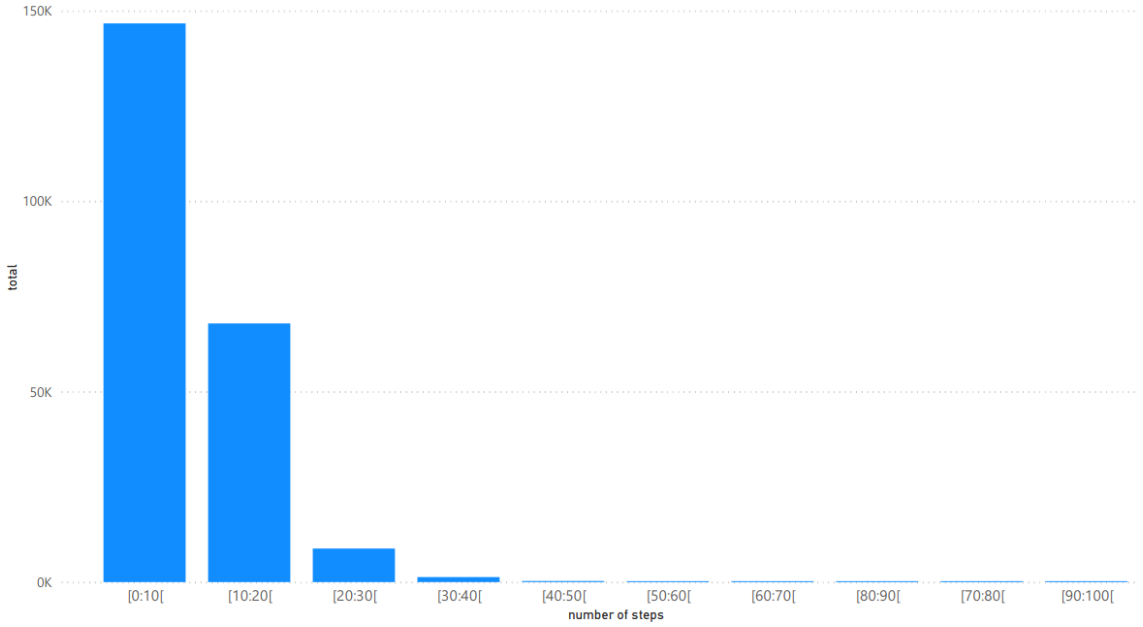

Figure 3.2: Total number of recipes by number of steps in the instructions to complete the food recipe

Number of ingredients

The *number of ingredients* attribute has a few examples which were defined as outliers. The outliers here were recipes with duplicate information or a very worldly ingredient description, in a sense the same problem about the user's upload of recipes. It was only considered recipes with less or equal to 18 ingredients. The outliers detected ascended to 3678 samples corresponding to 1,58% of the dataset. For example, instead of register only *salt*, the user choose *fine salt*. The dataset has recipes with less than seventeen ingredients. Figure 3.3 gives an insight into the number of ingredients per recipe. Three-quarters of all recipes require at least or less than ten ingredients. The majority of recipes requires between four and ten ingredients.

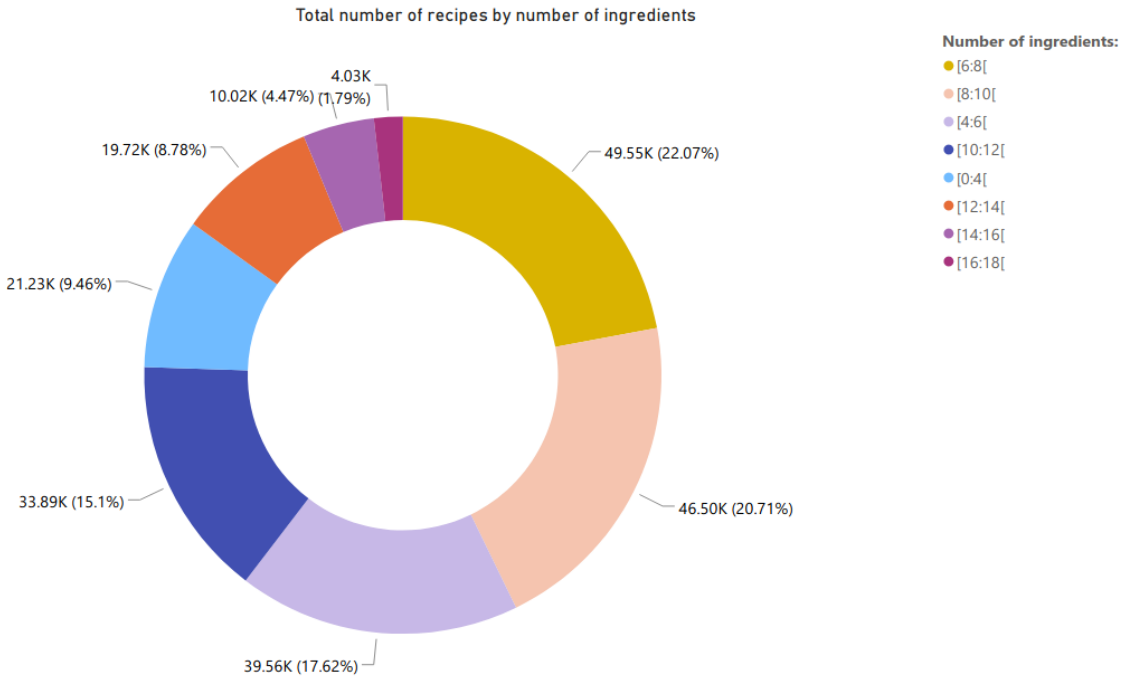

Figure 3.3: Total number of recipes by number of ingredients used

Analysing each ingredient's frequency on Figure 3.4 the ten most common ingredients are salt, butter, sugar, onion, water, eggs, olive oil, flour, milk and garlic cloves. The ingredients exploration revealed 14719 different ingredients.

Tag

Attribute tag is essential to distinguish similar recipes and group recipes by a determine characteristics. Users could make 445 tags to define recipes, with tags being used less than five different recipes. Figure 3.5 shows how they are distributed. Preparation,time-to-make, course dietary, main-ingredient, easy, occasion, cuisine, low-in-something, number-of-servings and main-dish are the leading ones and; after, the total decreases at a steady pace.

Figure 3.4: Top 100 most common ingredients in recipes

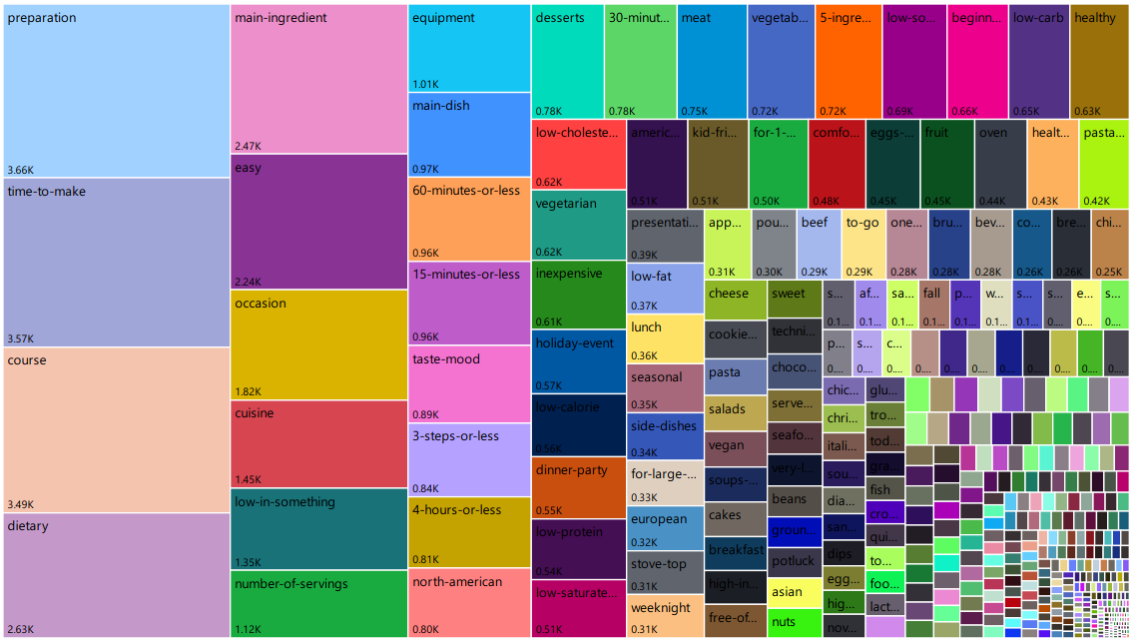

Figure 3.5: The sixty most common tags used for descriptive recipes.

It is possible to find exclusive tags easily and more descriptive: *60 minute-or-less* and *15 minutes-or-less*; *meat*, *vegetable* and *desserts*; or even tags that are more restrictive than the general ones: *low-in something* and *low-sodium*, *low-carb*, *low-cholesterol*.

Rating

The files corresponding to the rating given by the user on an item make it possible to analyse the user rating behaviour. A high majority of positive ratings globally composes the dataset. Users give a five-point to 530417 recipes, a four-point rating to 131846 recipes, while rating equal to three, two and one correspond to 27058, 7336 and 3722 recipes, respectively. Rating zero was given to 18000 recipes. This strong tendency is shown on the mean value: 4.41. The submissions' temporal interval is eighteen years, going from February 25, 2002, to December 19, 2018. The temporal length made it possible to have 1153 users who rated more than one hundred recipes.

Interactions

On the interactions files, it is essential to analyse if there is a small percentage of recipes that can concentrate a high volume of ratings and, also, if there is the quantity of recipes with few or without ratings. In other words, when considering users patterns is fundamental to explore popularity patterns. As shown on Figure 3.6 almost half of the dataset is compose by unique interactions, meaning a recipe with only one rating. It is expected a sparse matrix generation and therefore a significantly increase in memory and processing time. Also, it s a characteristic that will effect the algorithm's behaviour.

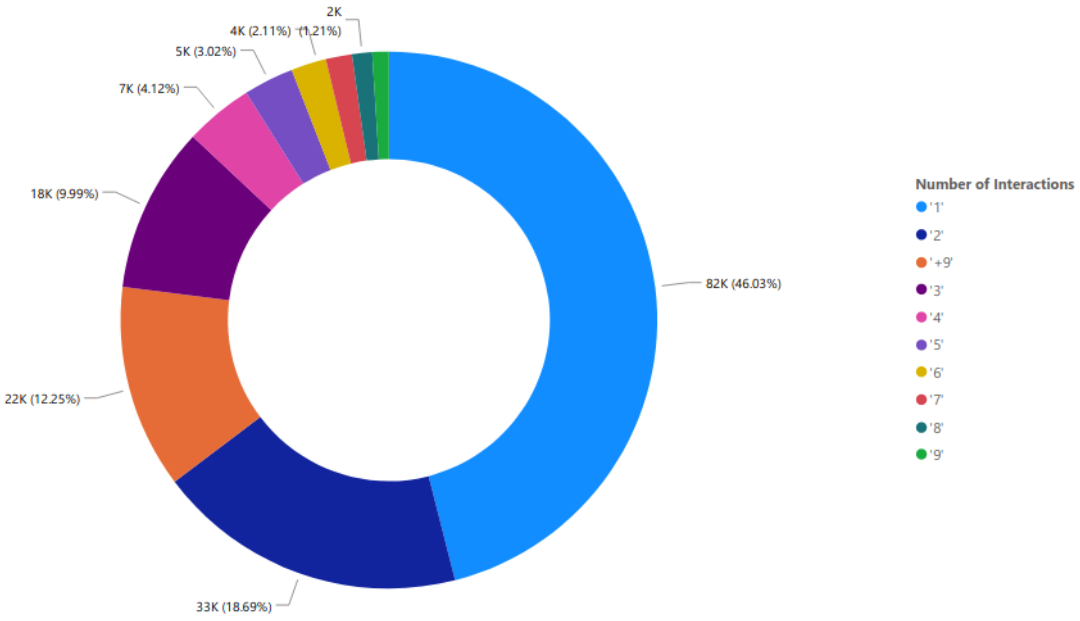

Figure 3.6: Number of interactions (a rating given by the user) for each recipe.

Chapter 4

Implementation

This chapter demonstrates the methodology used to validate the general approach on how to construct a model, that can predict whether a user u has a potential interest in an unknown item v, given a piece of information structured by a knowledge graph and the application of a neural network. Our methodology implemented well-known collaborative methods and a knowledge graph convolutional network with three distinct aggregation methods to identify the outcome and effects in the use of knowledge graphs in a recipe recommendation systems. Thus, Section 4.1 presents four collaborative methods: *SVD*, *Normal Predictor*, *KNNWithZScore*, *BaselineOnly* and *CoClustering*. Its results will be the basis for comparison with the implementation of the knowledge graph convolution network, described on Section 4.2, that uses three ways of feature aggregation: sum, concatenation and neighbourhood. Section 4.4 describe the metrics to evaluate the results obtained.

In order to make reading easier, a list of terms used frequently in the writing of this chapter is presented:

- $\hat{r_{ui}}$ is the prediction given for a user-item pair;
- μ is the mean of all ratings;
- μ_u is the mean of all ratings given by user u;
- b_u is the bias for user u;
- b_i is the bias for item i;
- $q_i^T p_u$ is equivalent to Probabilistic Matrix Factorisation algorithm, with q_i is the item factors and p_u the user factors;
- r_{ui} is the true rating of user u for item i;
- R_{train} is the training set;
- $N_i^k(u)$ the k nearest neighbours of user u that have rated item i;

4.1 Collaborative Filtering Methods

The idea of all methods in collaborative filtering is making a prediction or a set of predictions about one user's interests based on the interests of many other users. The explicit rating data is the main feature in this recipe dataset, so the objective is translated into using the pure rating data to learn a model to make predictions.

The python framework tool, *Surprise* [28] was chosen considering it is a specialised tool to build and analyse a recommendation system that deals with explicit rating data. This research uses four methods: *SVD*, *Normal Predictor*, *KNNWithZScore*, *BaselineOnly* and *CoClustering*.

The *SVD* algorithm, or Singular Value Decomposition, is a factorisation of the user-item matrix into three matrices, using algebra and linear operations such as eigenvalue decomposition matrix. When baselines are not used, this is equivalent to matrix factorisation [43]. The prediction rating for a given user-time pair is in Equation 4.1.

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \tag{4.1}$$

Normal Predictor algorithm predicts a random rating based on the distribution of the training set, which is assumed to be a normal distribution. The prediction $\hat{r}_{ui}$ is generated from a normal distribution $\mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$ where $\hat{\mu}$ and $\hat{\sigma}^2$ are estimated from the training data using Maximum Likelihood Estimation, according to Equations 4.2 and 4.3.

$$\hat{\mu} = \frac{1}{|\mathrm{R}_{\mathrm{train}}|} \sum_{r_{ui} \in \mathrm{R}_{\mathrm{train}}} r_{ui} \tag{4.2}$$

$$\hat{\sigma} = \sqrt{\sum_{r_{ui} \in \mathrm{R}_{\mathrm{train}}} \frac{(r_{ui} - \mu)^2}{|\mathrm{R}_{\mathrm{train}}|}} \tag{4.3}$$

KNN with Z-score algorithm derived from a primary nearest neighbours approaches that consider the z-score normalisation of data in each user. The prediction rating for a given pair user-time is in Equation 4.4.

$$\hat{r}_{ui} = \mu_u + \sigma_u \frac{\sum_{v \in N_i^k(u)} sim(u,v) \cdot (r_{ui} - \mu_v)/\sigma_v}{\sum_{v \in N_i^k(u)} sim(u,v)} \tag{4.4}$$

BaselineOnly algorithm works by predicting the baseline estimate for a given user and item. The prediction rating for a given user-time pair is in Equation 4.5.

$$\hat{r}_{ui} = v_{ui} = \mu + b_u + b_i \tag{4.5}$$

CoClustering is also called bi-clustering, where the goal is to generate co-clusters, a subset of rows that exhibit similar behaviour across a subset of columns, or vice versa. In other words, given some matrix A, we want to cluster rows of A and columns of A simultaneously. For our research,

this matrix will be a user-item matrix that will generate groups of related users. The prediction rating for a given user-time pair is in Equation 4.6.

$$\hat{r}_{ui} = \overline{C_{ui}} + (\mu_u - \overline{C_u}) + (\mu_i - \overline{C_i}) \tag{4.6}$$

All algorithms presented so far are implemented with cross-validation technique to prevent over-fitting or selection bias. The *Kfold* cross-validation technique train a model using the subset of the dataset and, then, evaluate it using the complementary subset of the data. In more detail, standard k-fold cross-validation makes *k* data subsets (folds) and, after, iteratively trains the algorithm on *k-1* folds while using the remaining fold as the test set (called the "holdout fold") as shown in Figure .

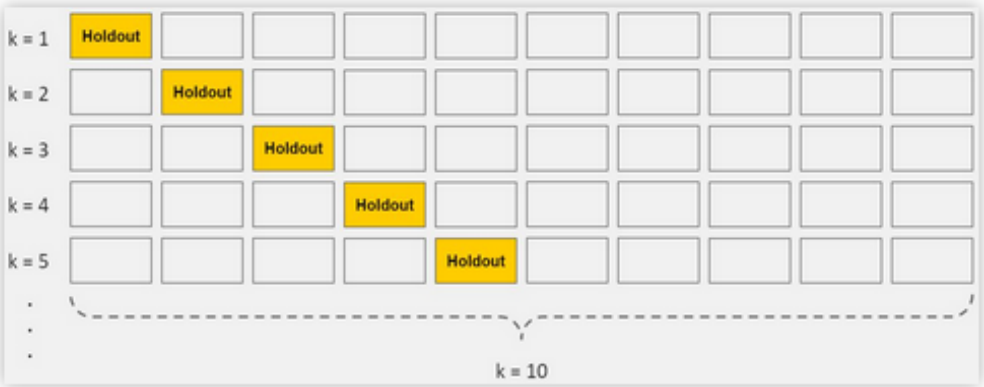

Figure 4.1: K-Fold Cross Validation

Five rounds of cross-validation are executed in different subsets of the dataset, and the validation results are combined in an estimate of the model's predictive performance. A selection of parameters for each algorithm is the result of the tuning process. The final parameters used will be specified for each algorithm. The implementation of random seed value has a crucial concern to have reproducible results and have consistency between experimental runs, as some cross validators iterators used it to generate subsets of the dataset.

The initial dataset has three files with user-item interactions. The three files were merged into one file for consistency in the cross-validation process.

4.2 Knowledge Graph Convolutional Network

This section gives an insight into the implementation of the knowledge graph convolutional network specific to the recipe dataset. The implementation is exposed in two complementary sections, as in both are some decisions that need to be explained. Section 4.2.1 explains the steps taken to build a knowledge graph given the recipes dataset. Section 4.2.2 explains the neural network implementation adapts to knowledge graphs with Wang et al. approach [64].

4.2.1 Knowledge Graph

The knowledge graph's implementation happens in two stages: first, define the ontology and build the knowledge graph and second, preparing all data for the graph convolution network.

The use of cooking ontology [44], food ontology [8] or recipe ontology, was considered however; given the dataset and objectives of this research, it is only considered a subset of the relationships present in the previous ontologies. The approach considered to construct the ontology is based on Paulheim [48] and Duan et al. [13]:

1. A knowledge graph defines real-world entities and their relations, organised in a graph;
2. A knowledge graph represents classes and relations of entities in a schema;
3. A knowledge graph allows entities linking with other entities;
4. A knowledge graph covers various popular domains.

A knowledge graph building takes three steps: entity extraction, relation extraction, attribute extraction. Given the dataset information, user and recipe are entities. Tag and ingredient are also entities and are connected to recipes. The knowledge graph relations are *hasTag*, *hasIngredient*, *hasPositiveRating* and *hasNumberOf Steps*. The relation *hasPositiveRating* is a translation of the user-item interactions where the user had given a rating of 4 or 5. The remains ratings were considered negative assessments and not considered.

The first knowledge graph is represented by relations in the following structure *(head entity, relation, tail entity)*. Entity disambiguation is necessary to ensure evaluation of quality and so, every entity has a unique identifier. An entity is an instance of a user, a recipe, a tag, an ingredient, a nutrition value and a number of steps. To summarise, the knowledge graph is characterised by the following relations:

- (user, *hasPositiveRating*, recipe)
- (recipe, *hasTag*, tag)
- (recipe, *hasIngredient*, ingredient)
- (recipe, *hasCookingTime*, cooking time)
- (recipe, *hasNumberOfSteps*, number of steps)

The most challenging in applying knowledge graph to recommendation systems is the high dimensionality and the heterogeneity of information. An example of high dimensionality is a recipe with a high number of ratings that in the graph will be a highly connected node. On the side of the heterogeneity, an example is a recipe with a considerable high number of tags and those also shared with other recipes. Storing complex and interrelated data in a knowledge graph to use in machine learning, is a complex task because the context of each data point means a high quantity of information. Existing machine learning techniques rely upon the existence of an input vector

for each example. Creating such a vector to represent a node in a knowledge graph is non-trivial. The problem-solving approach taken was using knowledge graph embedding methods, which map entities and relations to low-dimensional representation vectors, also shared in researches done by Huang et al. [27], Wang et al. [65] and Zhang et al. [68].

The *Tensor Flow* [1] library contains data structures for multi-dimensional tensors and respective mathematical operations. In our research, it is the package responsible for embedding the users, entities and relations. These graph embeddings will make it possible to do machine learning workflows and deterministic analytic workflows discussed in the next section.

4.2.2 Knowledge Graph-based Convolutional Network

The Knowledge Graph-based convolutional network implements an extended class from class *torch.nn.Module*, defined in *PyTorch* library. Class *torch.nn.Module* is a base class for all neural network modules. The *nn* modules provide a way to build and train deep network. In that way, a method forward was create to compute the network output and the configuration of different trainable layers, the fully connected layers, is accomplished with *torch.nn.Linear*.

Each graph node produces an aggregation of neighbourhood information, which includes the information relatively to relations and entities. Relations which translate relation scores are normalised with a softmax function. All features obtained are processed in each entity, weights are calculated depending on each score and therefore obtain a neighbourhood representation. This input representation supplies the layers of the neural network. The output v^u, layer's output and u, the user representation, will be the parameters for one of the three possible final aggregators: sum, concatenation and neighbourhood. The aggregators use the open-source library *Tensor flow* [1] to perform the dropout of outputs, a common regularisation done in neural networks layers, and to compute matrix operations for each aggregator. The propagation rule for each aggregator is defined in Equation 4.7, 4.8 and 4.9, with the corresponding notation:

- W is the matrix of weights;
- $v^u_{S(v)}$ is neighbourhood representation of the recipe v for the user u;
- b is the bias;
- v is the recipe representation;
- σ is the activation function *ReLu*.

$$Sum = \sigma(\mathrm{W} \cdot (v + v^u_{S(v)}) + b) \tag{4.7}$$

$$Concat = \sigma(\mathrm{W} \cdot concat(v, v^u_{S(v)}) + b) \tag{4.8}$$

$$Neighbourhood = \sigma(\mathrm{W} \cdot v^u_{S(v)} + b) \tag{4.9}$$

Finally, to reach a prediction rating the KGCN will have has for each user u we will calculate a score for a given relation r belong to user u. The recipe representation from the layer's output with the user representation is an input for the sigmoid function giving us a final prediction rating.

Finally, the dataset has divided 70% is for training, 15% for testing and 15% for results validation.

4.3 Complementary Experiments

After the implementation of non-graph methods and knowledge graph-based convolutional network method, two behaviours are possible to analyse and observed to go further in our research.

The first one, records results obtain from a subset of the dataset that contains only recipes with equal or greater than ten interactions, that is ten ratings given by different users to a recipe. The dataset will be reduced to 12897 recipes and 336094 interactions. In percentage these numbers translates to 29,75% of the total interactions data and 7,23% of the total recipes data. The objective is to reach a conclusion about the behaviour in dealing with a huge amount of concentrate ratings, instead of a sparse dataset.

The second complementary experiment will try to conclude if the results can improve when adding new relations to the knowledge graph. The knowledge graph-based convolutional network will run with a knowledge graph with only one relations *hasPositiveRating*, and thus a less dense knowledge graph with a fewer connections and less quantity of information.

4.4 Metrics

In information systems, a set of metrics have been used to evaluate recommendation systems results. However, considering that this research has an offline experimental setup, focuses on a prediction task and is evaluated on these predictions' accuracy; the evaluation metric chosen is the Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE). The main difference between the two metrics is the contribution of individual error values to the final result. These two metrics distinctiveness are the consideration of the absolute error for RMSE while MAE takes into consideration the measure of deviation. Both use the predicted rating for the user-item pair, P_{ui}, the rating given by user u to the item i, r_{ui}, the item's total number of ratings, N as in Equation 4.10 and 4.11.

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(p_{u,i} - r_{u,i})^2} \tag{4.10}$$

$$MAE = \frac{1}{N}\sum_{u,i}|p_{u,i} - r_{u,i}| \tag{4.11}$$

The RMSE gives an understanding of recommendation accuracy. RMSE increases with the variance of the frequency distribution of error magnitudes and give more importance to large

errors. Same goes for MAE, but it measures the average magnitude of the errors in a set of predictions, without considering their direction. In both measurements, the lower the value, the better.

Chapter 5

Results

All techniques implemented were ran on the same machine with the following specifications: Intel E5-2640 CPUs, 64GB of DDR3 RAM clocked at 1333MHz, QDR Qlogic QLE7340 IB HCA with Broadcom Corporation NetXtreme BCM5720 NICs. Table 5.1 shows the final results. Each model's final value is the result of optimisations obtained from consecutive runs with different hyper parameters and also with the help of *GridSearchCV* class, a *Tensor Flow* class specific for optimisations.

BaseLineOnly algorithm uses Alternating Least Squares (ALS) to do the baseline computation. Regularisation parameter for items and regularisation parameter for users is set at 10, 15 respectively. The optimisation set 10 iterations as the optimal number.

SVG algorithm is set to the following hyper parameters: 0,04 for regularisation parameter of the cost function; 0,005 for the learning rate; 15 iterations; 0,1 for standard deviation of the normal distribution for factor vectors initialisation and 0,02 for the regularisation term for all parameters. Number of factors was set to 100 and the number of iterations is set to 15.

CoClustering find the best performance when the number of iteration of the optimisation loop is 5, the number of user clusters is 3 and number of item clusters is also 3.

Knn with Z score find the best performance when the number of neighbours to take into account for aggregation is 10, and the minimum number of neighbours to take into account for aggregation is 12. The mean squared difference similarity between all pairs of users is the similarity used to estimate a rating.

KGCN find the best performance when the neighbour sampling size is 8, dimension of embeddings 64, depth of receptive field is 2, regulariser weight is 0,0002, learning rate is 0,002, batch size is 128 and epochs number is 50. These hyper-parameters were repeated in the three KGCN variants.

Model	RMSE	MAE
BaselineOnly	0,9310	0.5397
CoClustering	1,0112	0,5757
KnnWith ZScore	0,9973	0,5589
Normal Predictor	1,2033	0,7929
SVD	0,9203	0,5432
KGCN-sum	0,8620	0,4973
KGCN-concat	0,8612	0,5372
KGCN-neighbour	0,8495	0,4959

Table 5.1: RMSE and MAE values corresponding to the test dataset

From the set of non-graphs algorithms *SVD* and *BaselineOnly* has the ones with less error, being the *BaseLineOnly* more sensible to large prediction errors. *Normal Predictor* is the worst in predicting ratings, a result that was, in a way, expected.

Knowledge graph-based convolutional networks performed better in this research by reducing the prediction errors 0,08 points below compared to the best non-graph model. Inside the three different forms of aggregation, the neighbourhood aggregation is the one who performs better, followed close by the sum aggregation. Because the KGCN-neighbour works only with the neighbourhood entities, we can see the consequences of the lack of the entity features in itself when comparing with the KGCN-sum, showing a difference of 0,02 points in RMSE.

In other experiment the KCGN is ran with only part of the relationships defined previously. This knowledge graph is constructed with only one relationship, *hasPositiveRating*, and two types of entities, users and recipes, leading to the results presented in Table 5.2.

Model	RMSE	MAE
KGCN-Sum	0,9239	0.5529
KGCN-Concat	0,9248	0.5434
KGCN-neighbour	0,9201	0,5510

Table 5.2: RMSE and MAE values corresponding to the test dataset for a reduced knowledge graph

In fact, a KGCN with a reduced information performs worse when compared with the previous setup and shows results almost equal to the *SVD* algorithm in Table 5.1. This experiment shows that the adding of semantic information improves the results.

The last experience is performed only with a subset of the dataset. This experiment has the goal to check the behaviour when dealing with a scenario where all recipes have at least ten ratings. This setup is compared with the initial sparse dataset, composed by a high number of unique

ratings on recipes, which represents 46,03% of the whole initial dataset. Table 5.3 demonstrates the respective values.

Model	RMSE	MAE
BaselineOnly	0,4390	0,3897
CoClustering	0,4184	0,3541
KnnWith ZScore	0,4795	0,3589
Normal Predictor	0,4010	0,3241
SVD	0,3420	0,2848
KGCN-sum	0,3303	0,2933
KGCN-concat	0,3301	0,2854
KGCN-neighbour	0,3289	0,2758

Table 5.3: RMSE and MAE values corresponding to the test dataset with recipes that registered at least ten ratings

The consequence of dataset with at least ten ratings per recipe is a considerable decrease in the error measure in both RMSE and MAE. KGCN neighbour algorithm has the best performance followed closely by the other KGCN variants. From the KGCN variants, the Neighbour aggregator presents the lowest error between the three variants. In all experiences, Table 5.3 shows the lowest error variation and also shows a smaller interval between error measurements in comparison with the previous experiences.

Chapter 6

Conclusions and Future Work

This chapter provides a summary of the study developed, the conclusions regarding the comparison of both types of models, the implementation of the knowledge graph in a graph convolutional network and their applicability in recipes recommendation systems. Finally, the main contributions are enumerated and possible directions for future work are identified.

This dissertation focuses on developing a recommendation system that can help users identify the most suitable recipe for their current preference by predicting a rating for a recipe unknown to the user. The approach was to consider a recipe not as an individual item, but an item composed of attributes with different degrees of importance to the user. In this way, the knowledge graph was the technique to structure the recipe information in relationships and characteristics to help infer new knowledge patterns. The accomplishment of this objective leads to comparing non-graph hybrid methods with knowledge graph-based convolutional network. Their results leads us to conclude that recipes recommendation systems benefit from knowledge graph structures, applied in convolutional networks by reducing the error in the predicted ratings on sparse scenarios.

The complementary experiences allowed us to understand that, introducing more recipe information in the knowledge graph structure helps the model to reduce the error in the predicted ratings and mitigate the weakness related to sparse datasets. That is the difficulty to predict a rating for an item with few or nonexistent rates. The last experience, where only a recipe with more then ten ratings were considered, gives us an insight about the performance when presented with an ideal dataset for recommendation system; that is one where all items have a considerable number of ratings. The KGCN neighbourhood algorithm predicts with less error in this scenario but the other KGCN variants performed very similar. This scenario also demonstrates the smallest error measurement because there are numerous ratings between four and five, as the rating trend study revealed in Appendix A.3.

In all three experiences, the neighbour aggregator in the KGCN is consistently more accurate demonstrating the importance of the neighbourhood information to the predicting performance of the algorithm.

6.1 Main contributions

It is possible to identify two main contributions of this work in the information filtering field: first, the consideration of heterogeneity multi-criteria information in recipe recommendation systems, through the use of knowledge graphs and, secondly, a base comparison between the traditional models of recommendation systems and, models with a knowledge graph-based approach.

6.2 Directions for future work

New approaches to recommend systems are still emerging and recent researches show there is a set of possibilities, to analyse and to improve within the field of knowledge graph-based convolutional network.

Four possible directions for future research are identified.

First, designing a consistent and descriptive ontology for culinary domain and measuring its impact in the machine learning models. A more precise ontology could benefit the prediction result and also future machine learning models with an organise and structured workflow.

Secondly, researching how to map the impact and how to add the sentiment analysis from the reviews attribute to the knowledge graph. Knowledge graph ability to extract and infer wisdom from the facts were dependent of the relationships constructed. The sentiment analysis can lead to new relationships and assist in providing additional semantic information.

The third possible scenario for research is evaluating the knowledge graph-based in a real scenario, to identity the users behaviours in the set of recommendations given and analyse the accuracy of the predicting rating.

Finally, the forth direction is analysing how can a user ontology be added to the knowledge graph and, afterwards, identify if it can improve the overall recommendation system.

Appendix A

Data Exploration

This section exposes complementary visual representation of the dataset after removing the outliers identified. The box plots, on Section A.1 gave a better understanding of the data's variability across the recipe's attributes. Section A.2 displays the diagrams and charts to understand the correlation between attributes and study the dataset's characteristics. Finally, Section A.3 demonstrates produced charts that helped to identify the most potential attributes to be considered in the construction of the knowledge graph structure.

A.1 Recipes information visualisation

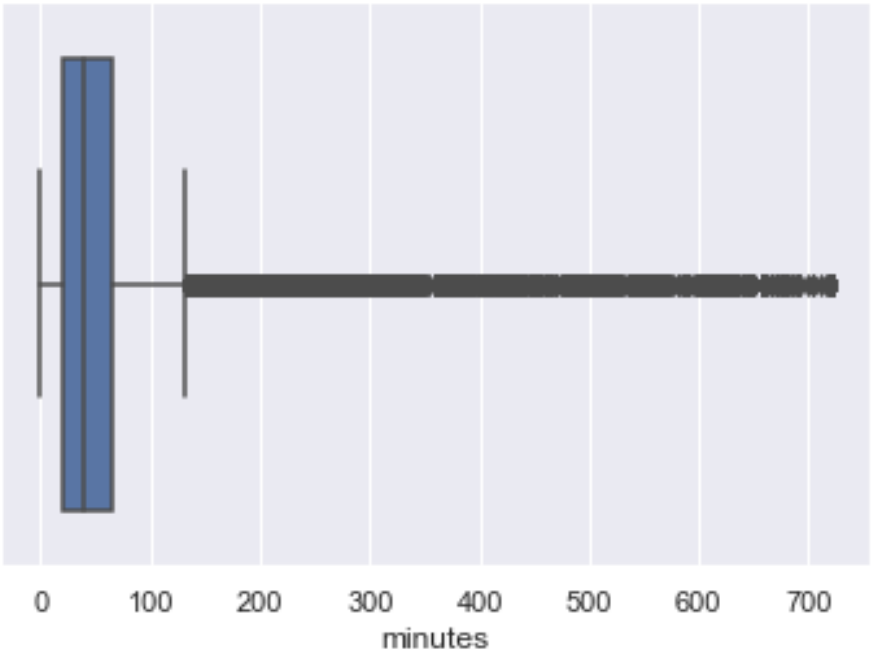

Figure A.1: Descriptive statistics using box plot method for attribute *minutes*, total time for recipe preparation

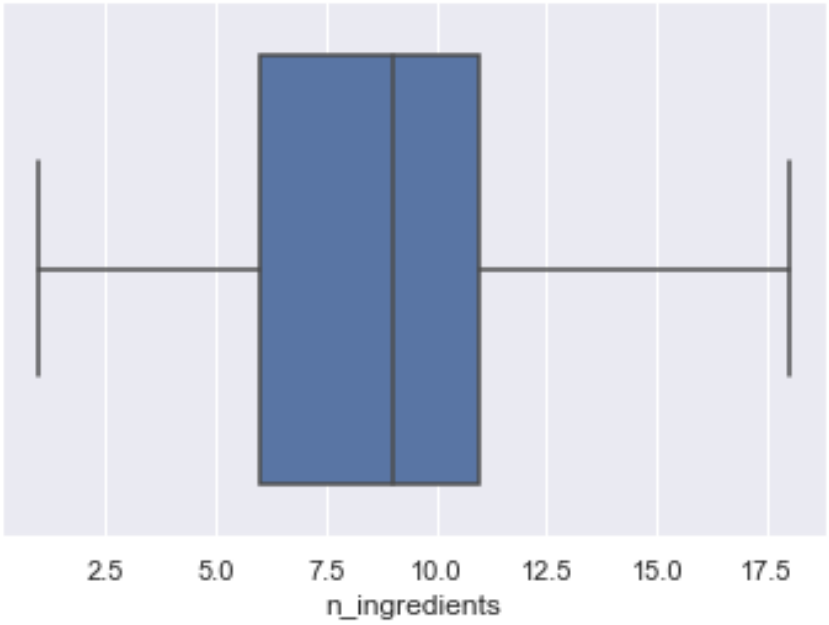

Figure A.2: Descriptive statistics using box plot method for attribute *n_ingredients*, total number of ingredients on the recipe.

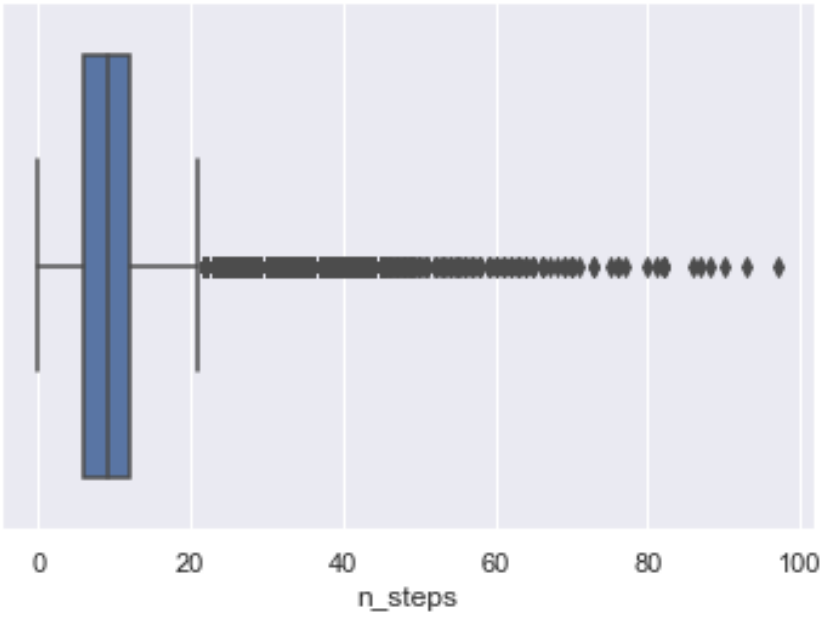

Figure A.3: Descriptive statistics using box plot method for attribute *n_steps*, total number of steps in recipes instructions.

A.2 Interactions Information Visualisation

A.2.1 File interactions_test.csv

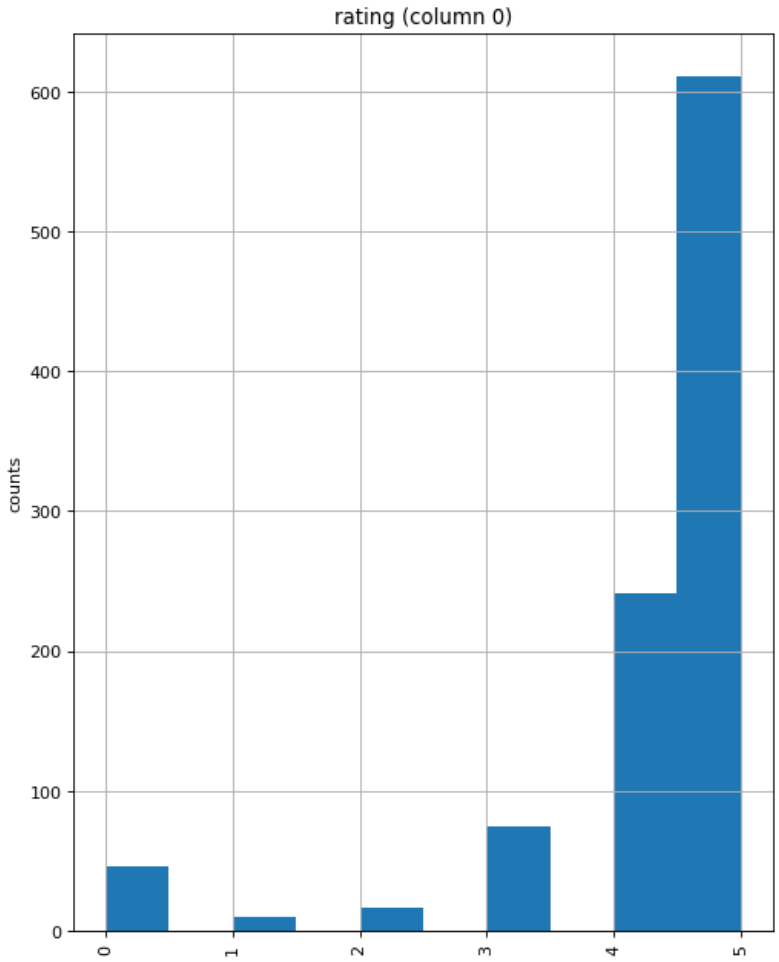

Figure A.4: Total number of recipes for each rating given by the user from 0 to 5 - test file

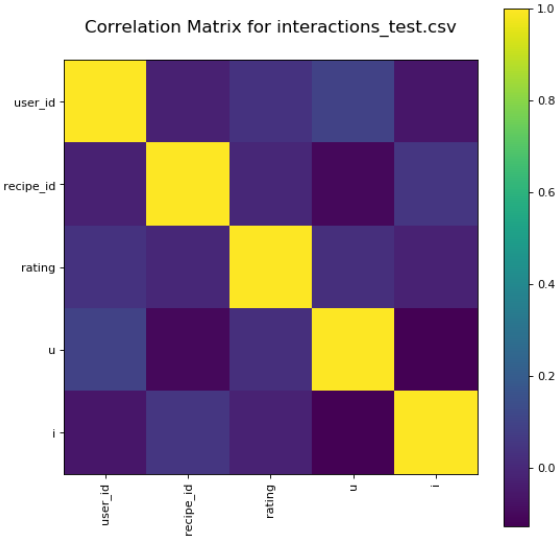

Figure A.5: Correlation matrix for attributes in interactions_test.csv

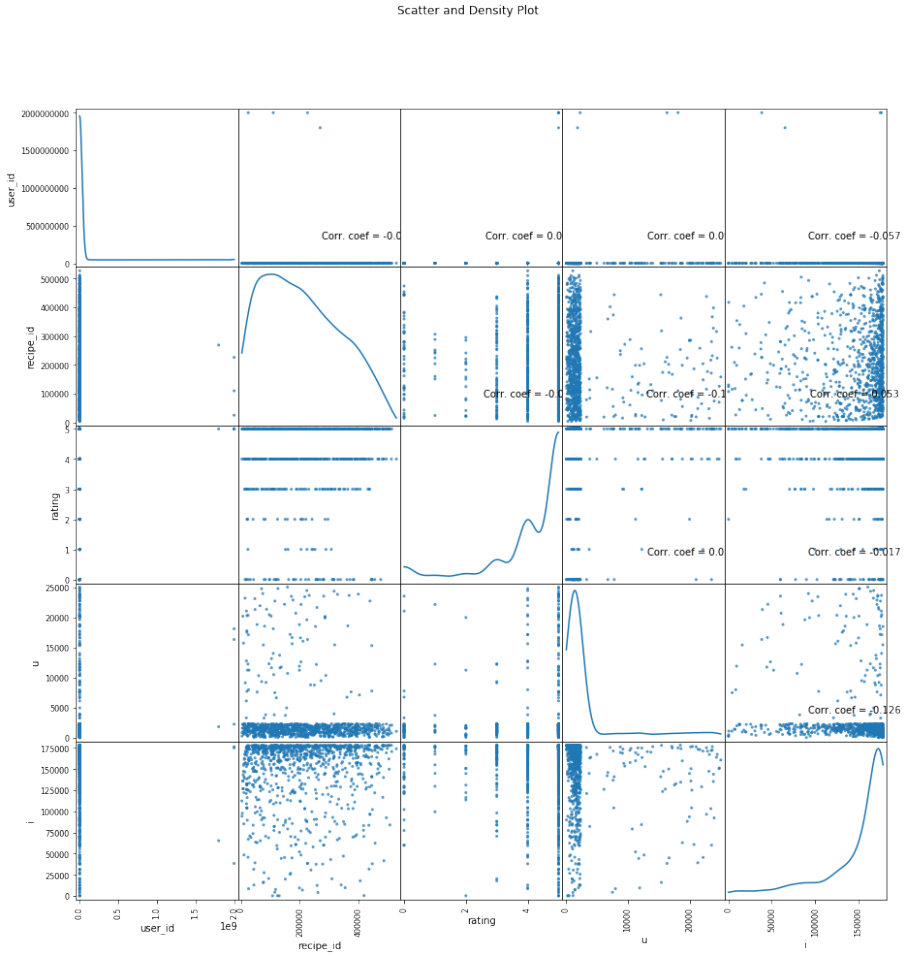

Figure A.6: Scatter and Density plot

A.2.2 File interactions_train.csv

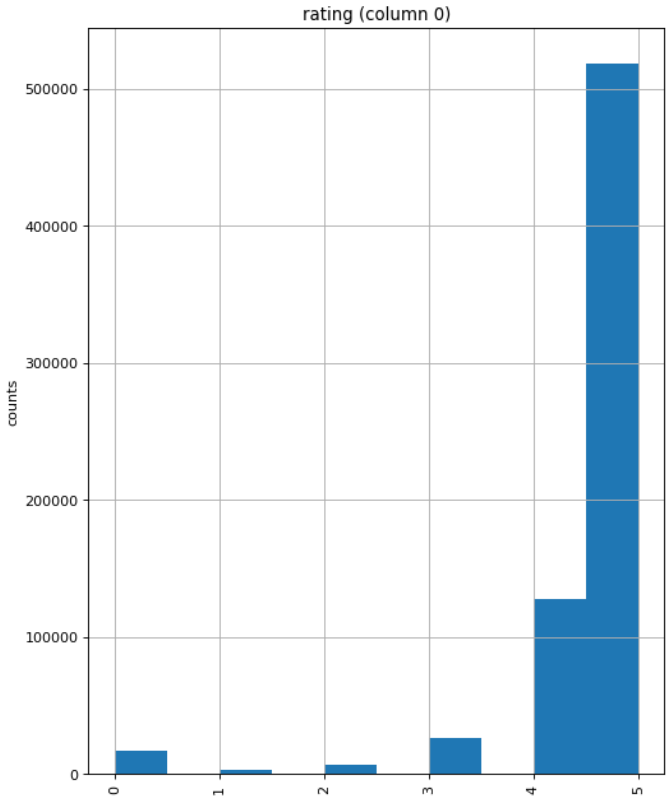

Figure A.7: Total number of recipes for each rating given by the user from 0 to 5 - train file

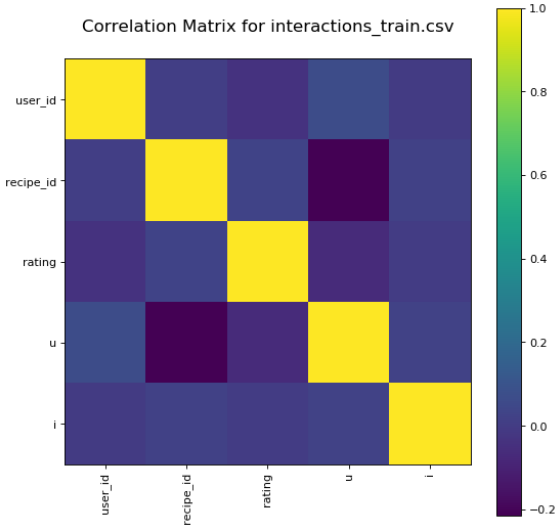

Figure A.8: Correlation matrix for train file

A.2.3 File interactions_validation.csv

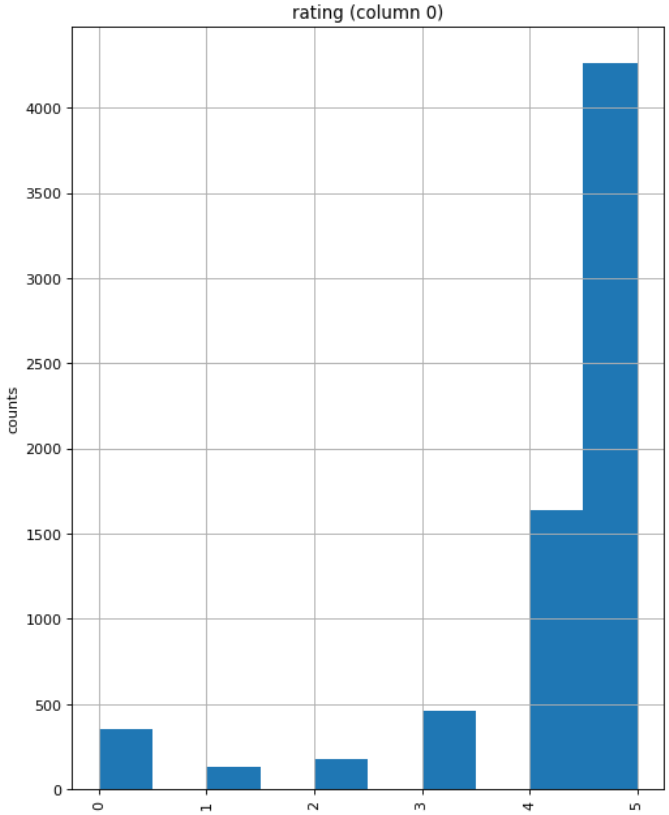

Figure A.9: Number of each rating given by the user from 0 to 5 - Validation file

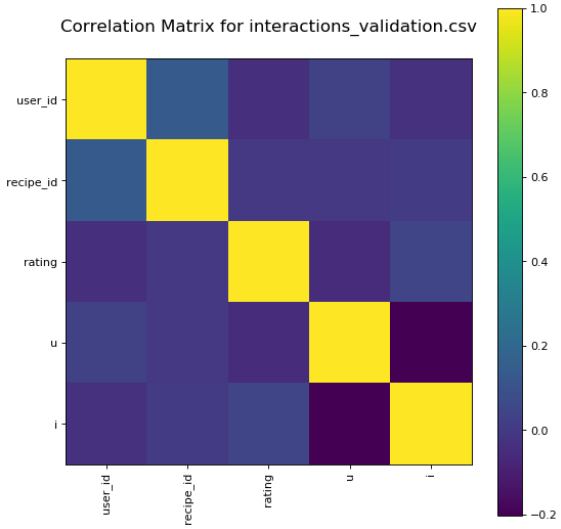

Figure A.10: Correlation matrix for validation filr

A.3 Ratings trends

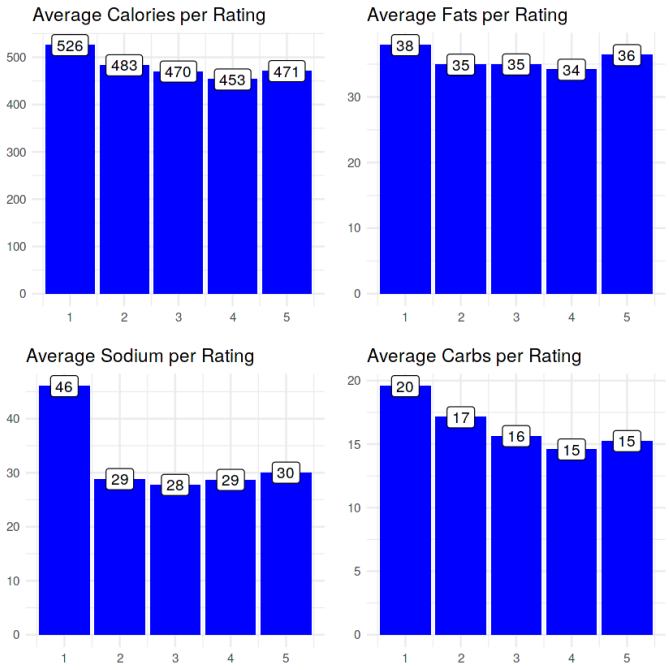

Figure A.11: Average number of nutritional values (y-axis) per rating value (x-axis)

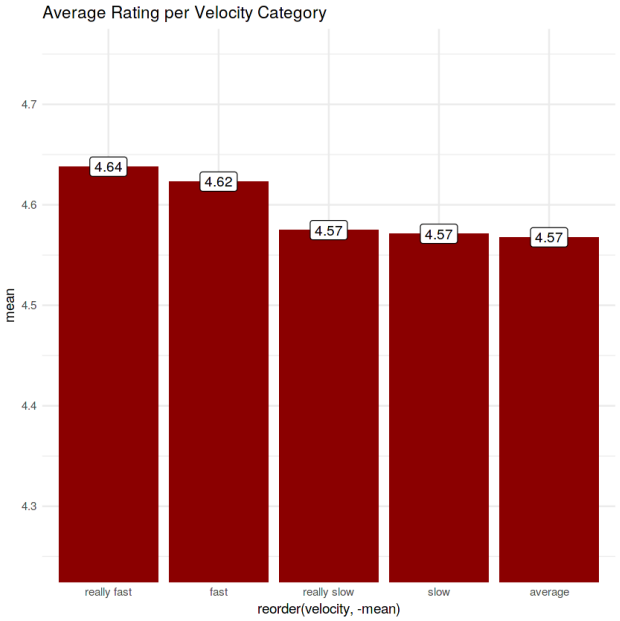

Figure A.12: Average rating by preparation time: "really fast" less than 10 minutes; "fast" between 10 and 20 minutes; "average" between 20 and 30 minutes; "slow" between 30 and 50 minutes and "really slow " greater than 50 minutes

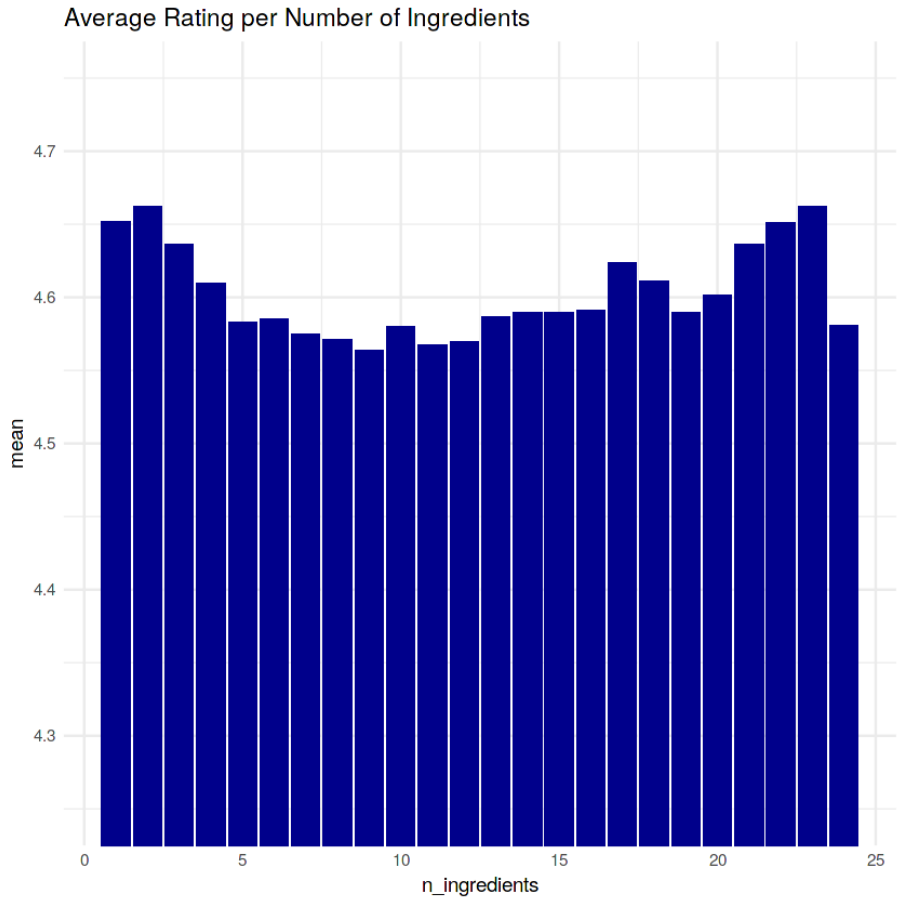

Figure A.13: Average rating (y-axis) per number of ingredients (x-axis).

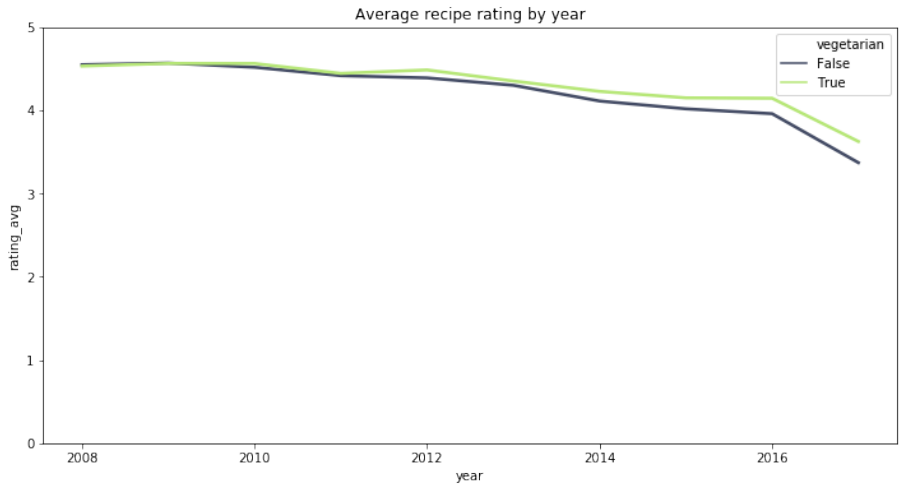

Figure A.14: Average recipe rating per year for vegetarian and non vegetarian recipes.

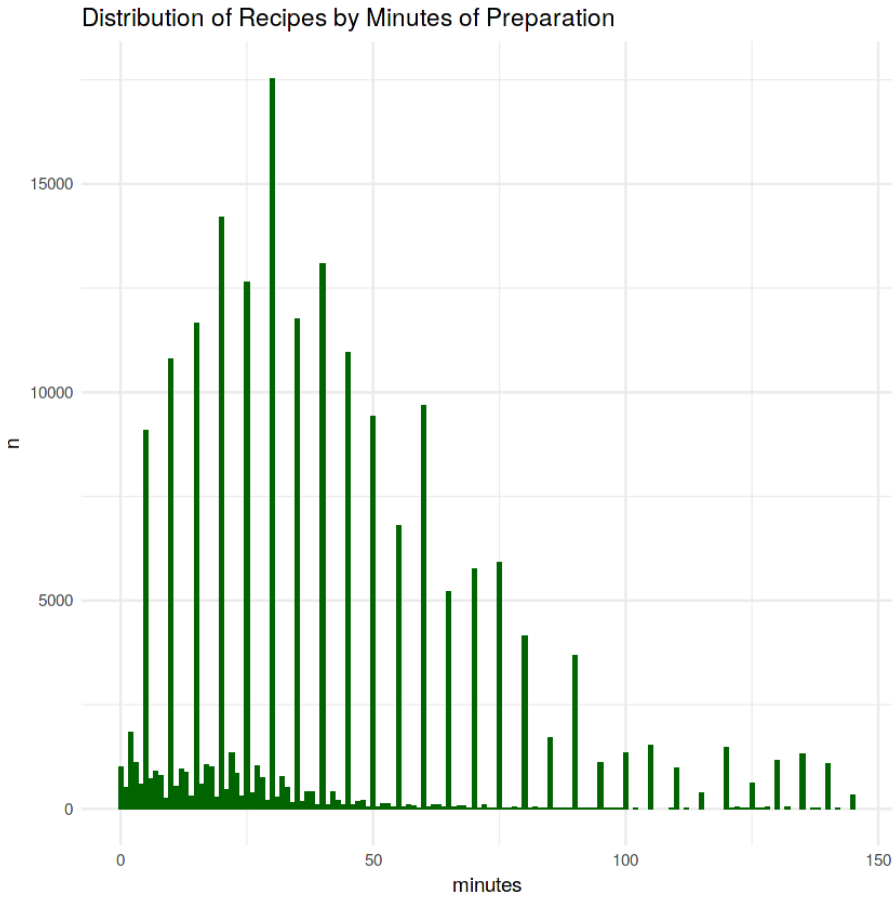

Figure A.15: Total of recipes by time of preparation

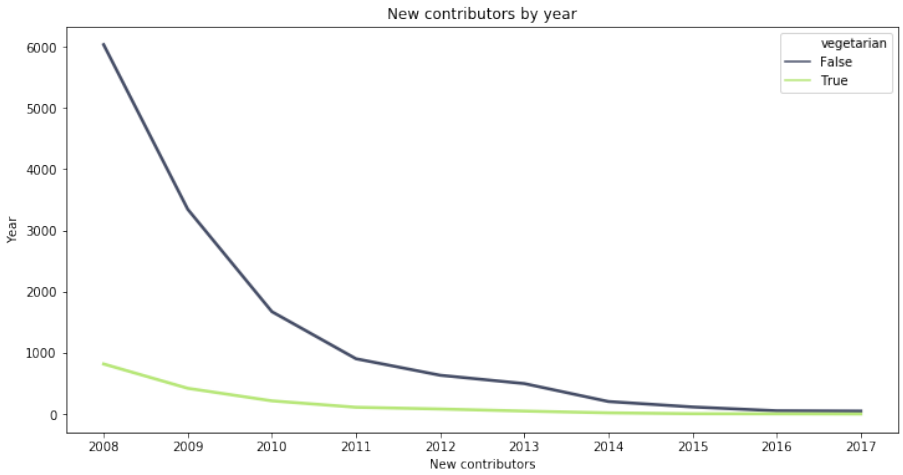

Figure A.16: New recipes uploads per year.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Johan Aberg. Dealing with malnutrition: A meal planning system for elderly. In *AAAI spring symposium: argumentation for consumers of healthcare*, pages 1–7, 2006.

[3] Gediminas Adomavicius and YoungOk Kwon. New recommendation techniques for multi-criteria rating systems. *IEEE Intelligent Systems*, 22(3):48–55, 2007.

[4] Gediminas Adomavicius, Nikos Manouselis, and YoungOk Kwon. Multi-criteria recommender systems. In *Recommender systems handbook*, pages 769–803. Springer, 2011.

[5] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

[6] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in neural information processing systems*, pages 1993–2001, 2016.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[8] BBC. Food ontology. https://www.bbc.co.uk/ontologies/fo, September 2020.

[9] Shlomo Berkovsky and Jill Freyne. Group-based recipe recommendations: analysis of data aggregation strategies. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 111–118, 2010.

[10] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[11] Erion Çano and Maurizio Morisio. Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis*, 21(6):1487–1524, 2017.

[12] Li Deng. Three classes of deep learning architectures and their applications: a tutorial survey. *APSIPA transactions on signal and information processing*, 2012.

[13] Yucong Duan, Lixu Shao, Gongzhu Hu, Zhangbing Zhou, Quan Zou, and Zhaoxin Lin. Specifying architecture of knowledge graph with data graph, information graph, knowledge graph and wisdom graph. In *2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA)*, pages 327–332. IEEE, 2017.

[14] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28:2224–2232, 2015.

[15] MA El-Dosuky, MZ Rashad, TT Hamza, and AH El-Bassiouny. Food recommendation using ontology and heuristics. In *International conference on advanced machine learning technologies and applications*, pages 423–429. Springer, 2012.

[16] Mehdi Elahi, Mouzhi Ge, Francesco Ricci, Ignacio Fernández-Tobías, Shlomo Berkovsky, and Massimo David. Interaction design in a mobile food recommender system. In *CEUR Workshop Proceedings*. CEUR-WS, 2015.

[17] Mehdi Elahi, Mouzhi Ge, Francesco Ricci, David Massimo, and Shlomo Berkovsky. Interactive food recommendation for groups. In *Recsys posters*. Citeseer, 2014.

[18] Alexander Felfernig, Martin Stettinger, Gerald Ninaus, Michael Jeran, Stefan Reiterer, Andreas A Falkner, Gerhard Leitner, and Juha Tiihonen. Towards open configuration. In *Configuration Workshop*, pages 89–94. Citeseer, 2014.

[19] Jill Freyne and Shlomo Berkovsky. Intelligent food planning: personalized recipe recommendation. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 321–324, 2010.

[20] Jill Freyne, Shlomo Berkovsky, and Gregory Smith. Recipe recommendation: accuracy and reasoning. In *International conference on user modeling, adaptation, and personalization*, pages 99–110. Springer, 2011.

[21] Anthony Goldbloom. Kaggle. https://www.kaggle.com/, September 2020.

[22] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. Leveraging multiviews of trust and similarity to enhance clustering-based recommender systems. *Knowledge-Based Systems*, 74:14–27, 2015.

[23] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. A novel recommendation model regularized with user trust and item ratings. *ieee transactions on knowledge and data engineering*, 28(7):1607–1620, 2016.

[24] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. A survey on knowledge graph-based recommender systems. *arXiv preprint arXiv:2003.00911*, 2020.

[25] Shweta Gupta and Vibhor Kant. A review and classification of multi-criteria recommender systems. In *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 1156–1162. IEEE, 2020.

[26] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[27] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. Improving sequential recommendation with knowledge-enhanced memory networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 505–514, 2018.

[28] Nicolas Hug. Surprise: A python library for recommender systems. *Journal of Open Source Software*, 5(52):2174, 2020.

[29] FO Isinkaye, YO Folajimi, and BA Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273, 2015.

[30] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 5308–5317, 2016.

[31] Stephen Kaufer. Tripadvisor. https://www.tripadvisor.com/, September 2020.

[32] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[33] Joseph A Konstan and John Riedl. Recommender systems: from algorithms to user experience. *User modeling and user-adapted interaction*, 22(1-2):101–123, 2012.

[34] Fang-Fei Kuo, Cheng-Te Li, Man-Kwan Shan, and Suh-Yin Lee. Intelligent menu planning: Recommending set of recipes by ingredients. In *Proceedings of the ACM multimedia 2012 workshop on Multimedia for cooking and eating activities*, pages 1–6, 2012.

[35] Qianyu Li, Xiaoli Tang, Tengyun Wang, Haizhi Yang, and Hengjie Song. Unifying task-oriented knowledge graph learning and recommendation. *IEEE Access*, 7:115816–115828, 2019.

[36] Shuyang Li. Food.com recipes and interactions. https://www.kaggle.com/shuyangli94/food-com-recipes-and-user-interactions, September 2020.

[37] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[38] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[39] Hanqing Lu, Chaochao Chen, Ming Kong, Hanyi Zhang, and Zhou Zhao. Social recommendation via multi-view user preference learning. *Neurocomputing*, 216:61–71, 2016.

[40] Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian McAuley. Generating personalized recipes from historical user preferences. *arXiv preprint arXiv:1909.00105*, 2019.

[41] Nikos Manouselis and Constantina Costopoulou. Analysis and classification of multi-criteria recommender systems. *World Wide Web*, 10(4):415–441, 2007.

[42] Stefanie Mika. Challenges for nutrition recommender systems. In *Proceedings of the 2nd Workshop on Context Aware Intel. Assistance, Berlin, Germany*, pages 25–33. Citeseer, 2011.

[43] Andriy Mnih and Russ R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.

[44] Hidetsugu Nanba, Yoko Doi, Miho Tsujita, Toshiyuki Takezawa, and Kazutoshi Sumiya. Construction of a cooking ontology from cooking recipes and patents. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pages 507–516, 2014.

[45] Department of Computer Science and Engineering at the University of Minnesota. Grouplens. https://grouplens.org/datasets/movielens/, September 2020.

[46] Iván Palomares and Sergey V Kovalchuk. Multi-view data approaches in recommender systems: an overview. *Procedia Computer Science*, 119:30–41, 2017.

[47] Chenguang Pan and Wenxin Li. Research paper recommendation with topic analysis. In *2010 International Conference On Computer Design and Applications*, volume 4, pages V4–264. IEEE, 2010.

[48] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2017.

[49] Yanru Qu, Ting Bai, Weinan Zhang, Jianyun Nie, and Jian Tang. An end-to-end neighborhood-based interaction model for knowledge-enhanced recommendation. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*, pages 1–9, 2019.

[50] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[51] Xiao Sha, Zhu Sun, and Jie Zhang. Attentive knowledge graph embedding for personalized recommendation. *arXiv preprint arXiv:1910.08288*, 2019.

[52] Margaret K Snooks et al. *Health psychology: Biological, psychological, and sociocultural perspectives*. Jones & Bartlett Publishers, 2008.

[53] Janusz Sobecki, Emilia Babiak, and M Słanina. Application of hybrid recommendation in web-based cooking assistant. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 797–804. Springer, 2006.

[54] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment*, 4(11):992–1003, 2011.

[55] Martin Svensson, Jarmo Laaksolahti, Kristina Höök, and Annika Waern. A recipe based on-line food store. In *Proceedings of the 5th international conference on Intelligent user interfaces*, pages 260–263, 2000.

[56] Xiaoli Tang, Tengyun Wang, Haizhi Yang, and Hengjie Song. Akupm: Attention-enhanced knowledge-aware user preference model for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1891–1899, 2019.

[57] Discovery Food team. Food. https://www.food.com/, September 2020.

[58] Chun-Yuen Teng, Yu-Ru Lin, and Lada A Adamic. Recipe recommendation using ingredient networks. In *Proceedings of the 4th Annual ACM Web Science Conference*, pages 298–307, 2012.

[59] Thi Ngoc Trang Tran, Müslüm Atas, Alexander Felfernig, and Martin Stettinger. An overview of recommender systems in the healthy food domain. *Journal of Intelligent Information Systems*, 50(3):501–526, 2018.

[60] Tsuguya Ueta, Masashi Iwakami, and Takayuki Ito. A recipe recommendation system based on automatic nutrition information extraction. In *International Conference on Knowledge Science, Engineering and Management*, pages 79–90. Springer, 2011.

[61] Youri van Pinxteren, Gijs Geleijnse, and Paul Kamsteeg. Deriving a recipe similarity measure for recommending healthful meals. In *Proceedings of the 16th international conference on Intelligent user interfaces*, pages 105–114, 2011.

[62] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 417–426, 2018.

[63] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 968–977, 2019.

[64] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. Knowledge graph convolutional networks for recommender systems. In *The world wide web conference*, pages 3307–3313, 2019.

[65] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[66] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 950–958, 2019.

[67] Milan Zeleny. *Linear multiobjective programming*, volume 95. Springer Science & Business Media, 2012.

[68] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362, 2016.

[69] Yongfeng Zhang, Qingyao Ai, Xu Chen, and Pengfei Wang. Learning over knowledge-base embeddings for recommendation. *arXiv preprint arXiv:1803.06540*, 2018.

[70] Jun Zhao, Zhou Zhou, Ziyu Guan, Wei Zhao, Wei Ning, Guang Qiu, and Xiaofei He. Intentgc: a scalable graph convolution framework fusing heterogeneous information for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2347–2357, 2019.