Universidade de Aveiro
2023

Anaísa
Lucena Silva

Abordagens Computacionais para Gerar e Completar Receitas Culinárias

Computational Approaches for Food Recipe Generation and Completion

Universidade de Aveiro
2023

Anaísa
Lucena Silva

Abordagens Computacionais para Gerar e Completar Receitas Culinárias

Computational Approaches for Food Recipe Generation and Completion

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Computacional, realizada sob a orientação científica do Doutor António Luís Ferreira, Professor associado do Departamento de Física da Universidade de Aveiro.

This research received funding from European Regional Development Fund (ERDF), Centro2020 and Agência Nacional de Inovação through the projet "Cook4Me", CENTRO-01-0247-FEDER-070247.

agradecimentos / acknowledgements

First and foremost, I express my gratitude to my advisor, Professor António Luís Ferreira, whose support, motivation, and guidance have been instrumental throughout this journey.

I am also grateful for the support provided by Wish and Cook and the project cook4me. Specially, I extend my sincere appreciation to Professor Fernão Abreu, for his guidance, motivation, and willingness always help. Additionally, my thanks to Ana Sofia Freitas and Professor Carlos Azevedo.

Finally, I want to thank my mother and grandmother for their support and encouragement. I am also thankful for Guilherme, whose presence and advice proved invaluable throughout this process. At last, I want to express my gratitude to God, for guiding me through this journey.

Palavras Chave receitas culinárias, ingredientes, cadeias não-Markovianas, RBM, NMF.

Resumo

A indústria alimentar moderna depara-se com o aumento das alergias alimentares e da obesidade, havendo necessidade de personalização, e com a luta global contra a fome, sendo necessárias soluções inovadoras e que reduzam o desperdício. Este estudo explora receitas melhoradas por inteligência artificial, com o potencial para atender a estes desafios. Três métodos computacionais: uma cadeia não-Markoviana, uma máquina de Boltzmann restrita (RBM) e uma factorização de matriz não-negativa (NMF) foram utilizados para gerar e completar receitas. Estes métodos foram avaliados usando duas bases de dados distintas: uma mais pequena, que contém exclusivamente sopas, e uma maior e mais diversa, com receitas de várias partes do mundo.

Em relação ao desempenho dos métodos, o algorítmo de Markov criou muitas receitas repetidas, enquanto o RBM e o NMF geraram receitas diversas e únicas, o que se deve à tendência do algorítmo de Markov de sugerir principalmente os ingredientes mais comuns. Adicionalmente, o método de Markov oferece a vantagem de não requerer afinação. Por outro lado, o NMF exige o ajuste de um único parâmetro, e o RBM tem a desvantagem de necessitar o ajuste de quatro parâmetros. O RBM também preservou melhor o coeficiente de correlação de Pearson e a informação mútua originais entre pares de ingredientes que os métodos Markov e NMF. Isto foi demonstrado através de testes que envolveram o cálculo das distâncias euclidianas entre os valores calculados diretamente das bases de dados e aqueles obtidos a partir das receitas geradas pelos métodos, assim como a agregação hierárquica de ingredientes e a comparação entre as matrizes de correlação dos 50 ingredientes mais comuns. É de salientar que os métodos Markov e RBM evitaram com sucesso pares proibidos de ingredientes, enquanto o RBM e o NMF evitaram eficazmente combinações raras de ingredientes populares. Além disso, ao completar uma receita de teste, os três métodos forneceram sugestões com sentido.

A exploração de relações mais complexas entre os ingredientes, incluindo correlações que envolvam 3 ou 4 ingredientes, e a consideração de retorno dos utilizadores, anteveem-se como trabalho futuro promissor. Além do mais, a longo prazo, a integração das quantidades dos ingredientes e dos passos de preparação das receitas poderia aumentar significativamente a profundidade e a aplicabilidade destas receitas melhoradas por inteligência artificial.

Keywords food recipes, ingredients, non-Markovian chains, RBM, NMF.

Abstract

The modern food industry is confronted with the rise of food allergies and obesity, requiring personalization, and the global struggle against hunger, demanding innovative, waste-reducing solutions. This study explores AI-enhanced recipes, with the potential to address these challenges. Three computational methods: a non-Markovian chain, a Restricted Boltzmann Machine (RBM), and Non-negative Matrix Factorization (NMF) were employed to generate and complete recipes. These methods were evaluated using two distinct databases: a smaller one exclusively containing soups, and a larger, more diverse database, featuring recipes from around the world.

Regarding the performance of the methods, the Markov algorithm exhibited repetitive recipes, while the RBM and NMF generated diverse and unique recipes, which happened due to the tendency of the Markov algorithm to mainly suggest the most common ingredients. Furthermore, the Markov method offers the advantage of requiring no tuning. On the other hand, the NMF necessitates the tuning of a single parameter, and the RBM incurs in the disadvantage of requiring the fine-tuning of four parameters. The RBM also preserved the original Pearson correlation coefficient and mutual information between ingredient pairs better than both the Markov and NMF methods. This was demonstrated through tests involving the calculation of Euclidean distances between the values calculated from the databases and those obtained from the recipes generated by the methods, as well as performing hierarchical clustering and comparing the correlation matrices of the 50 most common ingredients. Notably, the Markov and RBM successfully avoided prohibited ingredient pairs, while the RBM and NMF effectively avoided rare combinations of popular ingredients. Moreover, when completing a test recipe, all three methods provided sound suggestions.

Looking forward, evaluating the methods by looking at more intricate ingredient relationships, such as correlations involving 3 or 4 ingredients, and incorporating user feedback, stand promising for future work. Moreover, in the long term, integrating ingredient quantities and cooking steps could significantly enhance the depth and applicability of these AI-enhanced recipes.

Contents

List of Acronyms

AI	Artificial Intelligence
AIV	Amazon Instant Video
CF	Collaborative Filtering
CNN	Convolutional Neural Network
DCNN	Deep Convolutional Neural Network
DGN	Decomposition Generation Network
KL	Kullback-Leibler
MAE	Mean Absolute Error
MCMC	Markov Chain Monte Carlo
ML	Machine Learning
MLE	Maximum Likelihood Estimation
NLP	Natural Language Processing
NMF	Non-negative Matrix Factorization
RBM	Restricted Boltzmann Machine
RDE-GAN	Recipe Disentangled Embedding Generative Adversial Network
RLS	Regularized Least Squares
RNG	Random Number Generator
RS	Recommendation System
SDG	Sustainable Development Goal
SVD	Singular Value Decomposition
WWW	World Wide Web

List of Figures

List of Tables

Chapter 1

Introduction

The food industry is showing a rising interest towards personalized nutrition, an approach to enhance health by tailoring dietary recommendations to individual needs, promoting well-being and preventing diseases effectively [1].

1.1 Motivation

As food allergies, obesity, and other food-related diseases continue to rise, the need for personalized nutrition grows ever more significant. Food allergies currently affect up to 10% of the general population [2] and obesity rates have nearly tripled between 1975 and 2016. According to the World Health Organization, in 2016, 39% of the world's adult population could be classified as overweight and 13% as obese [3].

Furthermore, in today's world a pressing issue of global concern is hunger. In 2022, approximately 735 million individuals struggled with hunger, while 2.4 billion people faced food insecurity [4]. Moreover, a substantial portion of the world's food supply, about 14%, is lost during production and distribution [5], with an additional 17% wasted mainly at the retail and consumer levels [6]. The convergence of these issues adds urgency to the search for innovative solutions that not only optimize global food systems, thereby reducing food waste, but also address both dietary and health-related challenges.

In the context of the Fourth Industrial Revolution, where advanced digital technologies are reshaping various industries [7], the culinary domain is also undergoing a significant transformation through, for example, the application of artificial intelligence (AI), a broad term referring to the computer-based modeling of intelligent behavior with minimal human intervention, to generate and enhance recipes. This innovative integration of AI and gastronomy aligns with the principles of Industry 4.0, emphasizing automation, data-driven insights, and collaborative innovation [8, 9].

AI-enhanced recipes have the potential to optimize food production, reduce food waste, and offer more nutritious and efficient meal choices, thereby contributing to the global efforts aimed at eradicating hunger and enhancing food security (Sustainable Development Goal 2 - Zero hunger) [10]. In addition to its direct alignment with the Sustainable Development Goals (SDGs), the use of AI to generate or complete recipes resonates with the concept of smart cities. As urban centers enhance efficiency and quality of life, intelligent culinary systems allow the city to foster a varied and sustainable food supply [11].

The utilization of AI-enhanced recipes can also assist in designing personalized and healthier meal plans, taking into account dietary preferences, allergies, and nutritional requirements, thereby promoting better eating habits and overall well-being (SDG 3 - Good health and well-being). Moreover, they can encourage the selection of sustainable food options and recipes incorporating locally sourced, seasonal, and environmentally friendly ingredients, thereby promoting responsible consumption and production patterns (SDG 12 - Responsible consumption and production) [10].

Anticipating the advent of the Fifth Industrial Revolution, characterized by an unprecedented level of automation, harmonious human–machine collaborations and sustainable practices [12], the role of AI in the culinary sphere gains even greater significance. As smart cities become a defining feature of the future, AI-enhanced recipes can promote sustainable consumption by optimizing ingredient utilization, minimizing food wastage, and providing tailored dietary suggestions (personalized nutrition). This convergence of technology and gastronomy underlines the transformative potential of AI in cultivating a more connected, efficient, and conscious society [11, 13].

Moreover, specially when paired with smart food services, AI-enhanced recipes offer economic benefits through cost-effective ingredient choices, as well as time efficiency, due to the optimization of processes. This allows a level of customization that caters to individual preferences and dietary needs, nutritional analysis for informed decision-making, and guidance for portion control to support healthier eating habits without having to cook or have a personal chef. By addressing these factors, AI-enhanced recipes enhance accessibility to meal plans and personalized nutrition, bringing about a more equitable and well-rounded approach to healthy eating. The benefits of AI-enhanced recipes are illustrated in Fig.1.1.

However, the current state of AI-enhanced recipes exhibits limitations in achieving the level of personalized nutrition found in human-prepared meals. This discrepancy primarily arises from the intricate nature of food pairing and the multifaceted realm of sensory perception that humans possess, where elements such as colors, texture, temperature, and sound can all contribute to the way individuals experience food [14].

As such, one of the challenges of generating AI recipes lies in comprehending the intricate interplay of flavors, textures, and aromas that unfold during the cooking

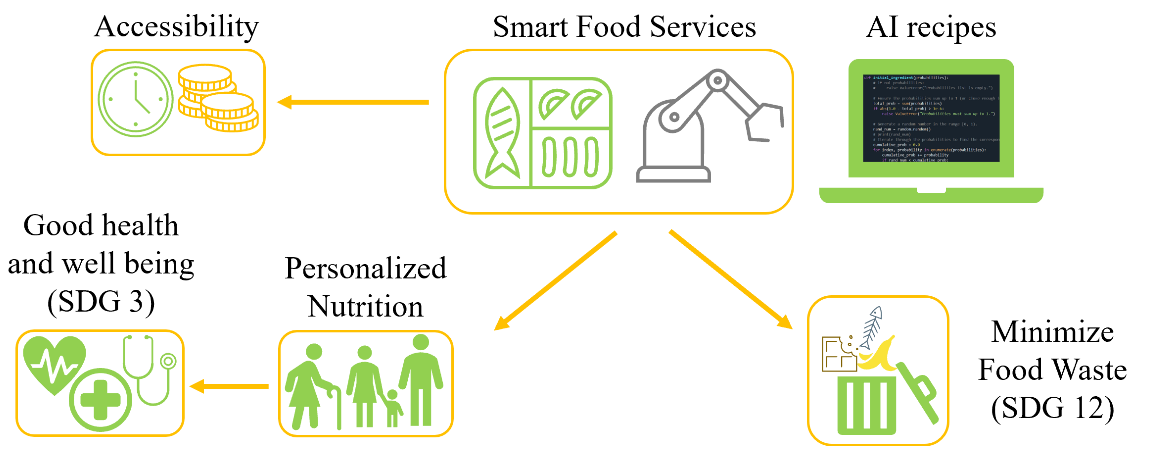


Figure 1.1: Schematic of the interrelationships between smart food services and AI recipes with accessibility, personalized nutrition and optimization of ingredient utilization.

process. While AI models have advanced in creating recipes and meal plans, they often lack an intuitive understanding of ingredient compatibility in a culinary context. Additionally, the data engineering aspect also presents challenges, such as standardizing recipe formats, cleansing and normalizing recipe data, and extracting ingredient and nutritional details. Moreover, scalability and performance are crucial concerns, especially as the volume of recipes in a management system grows, necessitating the system's ability to efficiently handle expanding data loads while ensuring rapid query responses [15].

Nevertheless, the future of AI-enhanced recipes holds promise. As ongoing research advances the understanding of culinary science and human nutrition, AI algorithms can continue to evolve, bridging the gap between convenience and personalized nutrition. By leveraging data from diverse sources, such as nutritional databases, sensory analyses, and individual dietary needs, AI-driven meal planning and enhanced recipes could eventually offer efficient and widely accessible options tailored to individual health goals.

By relying on objective data and scientific principles, AI-enhanced recipes could provide an approach to meal planning with decreased human bias, ensuring that dietary choices stem from accurate nutritional information rather than preconceived notions or cultural biases.

1.2 Objectives of the Work

The focus of this study centered around several objectives, all of which played a crucial role in achieving the primary aim of creating a reliable recipe generation and recipe completion framework and gaining a comprehensive grasp of the underlying patterns within culinary datasets.

Initially, the objective was to develop a versatile recipe generation and completion framework integrating Non-negative Matrix Factorization (NMF), Restricted Boltzmann Machines (RBM), and Markov chain based algorithms. This aimed at creating a computational model capable of generating or completing diverse and coherent recipes by learning from the underlying patterns within the training databases.

The evaluation of the computational time complexity of the implemented algorithms was the second objective, since ensuring swiftness in the recipe generation process is essential to the practical usability of the developed framework.

Furthermore, the algorithms' ability to capture intricate patterns within recipe datasets was assessed by exploring the relationships and hidden connections between ingredients, both within the datasets and the recipes generated by the algorithms. This exploration was conducted through in-depth analysis of mutual information and correlation coefficients, as well as performing hierarchical clustering.

These analyses provided valuable insights into the culinary domain, which was fundamental for a qualitative assessment of the generated and completed recipes. This qualitative assessment involved exploring a diversity of generated recipes to evaluate their coherence and relevance, as well as the completion of a generic recipe, ensuring both statistical accuracy and human appeal.

To accomplish these objectives, this study investigated the aforementioned computational algorithms and their practical implementations in the fields of recipe generation and completion. The following sections provide in-depth discussions on the theoretical foundations and state of the art, methodologies employed, and the results and analyses associated with each objective.

1.3 Publications

During the development of this study, a conference paper was published and presented, as shown below.

A. Lucena, A. S. Freitas, A. L. Ferreira, and F. Abreu, "Inspiration from systematic literature reviews to predict the future of food services in smart cities," in SMART 2023, The Twelfth International Conference on Smart Cities, Systems, Devices and Technologies, ThinkMind, IARIA, **2023**

This paper was recognized as one of the five "Best Papers" based on evaluations of the original submission, the camera-ready version, and the presentation at the conference. Consequently, it received an invitation to submit an extended article version to the IARIA International Journal On Advances in Intelligent Systems, scheduled for publication in volume 17, numbers 1&2 in **2024**.

Chapter 2

Computational Gastronomy: Concepts and State of the Art

Computational gastronomy is an emerging interdisciplinary field that combines elements of computer science, artificial intelligence, data science, and culinary data to explore and innovate in the realm of food and cooking [16]. In this chapter, the concepts of artificial intelligence and machine learning will be introduced and explained. This chapter delves into the recent significant advancements that have been achieved in its subfields, including enhanced food recognition technologies for ingredient and dish identification, the development of recipe recommendation systems (RS), methods for generating and refining recipes through recipe ideation, the exploration of food pairings, and the automation of recipe completion, as well as its current state of the art.

Next, the chapter addresses prediction models and recommendation systems relevant to this study, and found in the literature, encompassing Markov processes, restricted Boltzmann machines, and matrix factorization.

2.1 Artificial Intelligence and Machine Learning

In 1956, the concept of artificial intelligence emerged as a pioneering academic discipline. Coined by John McCarthy in 1955, AI was defined as the science and engineering devoted to crafting intelligent machines. McCarthy, a significant figure in AI's early development, collaborated with colleagues to establish the field at a 1956 Dartmouth College conference on artificial intelligence. This conference marked the official birth of AI as a scientific pursuit, setting the stage for its subsequent advancements [17].

Furthermore, AI is a broad term referring to the computer-based modeling of intelligent behavior with minimal human intervention. While it shares similarities with the task of using computers to comprehend human intelligence, AI is not restricted to biologically observable methods [17, 18]. In fact, AI goes beyond mere understanding, aiming to actively create intelligent entities, marking its ambitious pursuit within the sphere of technological intelligence advancement [19].

In 1959, IBM researcher Arthur Samuel introduced the term "machine learning" (ML), a subset of AI focused on developing algorithms and models that enable machines to autonomously improve their performance through learning from data and their environment, without explicit programming. Samuel's innovative work included programming computers to learn tasks like playing checkers [20]. Machine learning has since evolved into a versatile discipline with applications across diverse fields, including pattern recognition [21], computer vision [22], e-commerce [23], natural language processing [24], and medical applications [25].

At the core of ML is the utilization of data to train models. This data-driven approach involves providing a vast dataset to the system, enabling it to uncover underlying patterns and relationships. Through iterative adjustments of internal parameters, the ML model optimizes its performance over time. The ML process encompasses various techniques, including supervised, unsupervised, semi-supervised, and reinforcement learning [26, 27].

In supervised learning, the model is trained on labeled data, where the correct outcomes are provided, enabling the system to make predictions or classifications on new, unseen data (e.g., classification and regression). Unsupervised learning, on the other hand, deals with unlabeled data and focuses on finding inherent structures or patterns within the dataset (e.g., clustering, dimensionality reduction and association data mining) [26, 28]. Fig.2.1 shows various types of supervised and unsupervised learning. Semi-supervised learning involves a combination of labeled and unlabeled data (such as user product ratings). This type of ML harnesses the power of labeled data to extract insights from the unlabeled portion, analogous to how humans develop skills. Finally, reinforcement learning involves a feedback-driven process where the model learns to make sequential decisions to maximize a reward, similar to teaching animals through external cues [26, 27, 28].

In the field of using machine learning to create, improve, or complete recipes, unsupervised learning is employed. This technique helps uncover patterns among ingredients in a collection of recipes, enabling a better understanding of how they relate to each other.

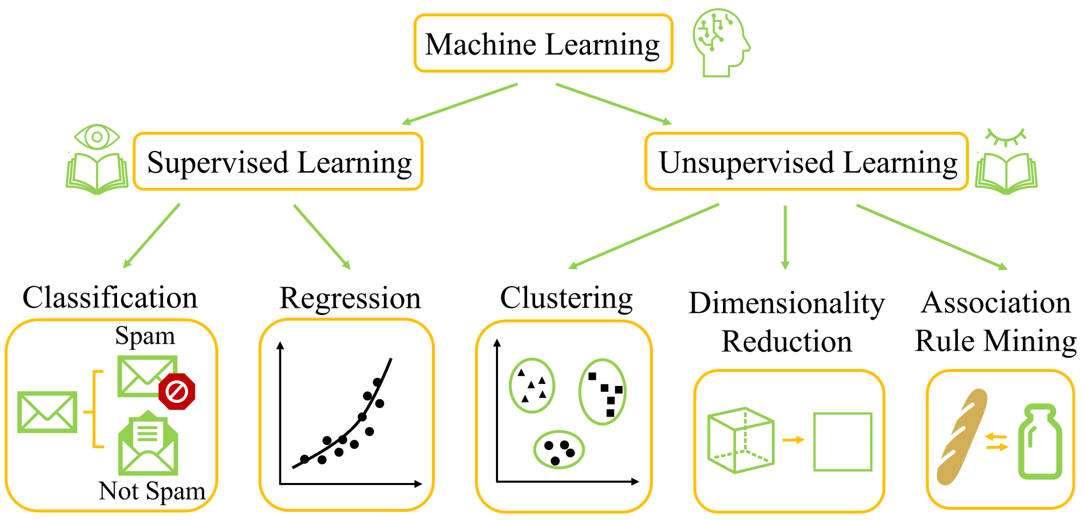


Figure 2.1: Schematic of various types of supervised and unsupervised learning.

2.2 AI Applications in Culinary Science

A recipe can be understood as an array of repeatable aspects of a dish, whose replication would deliver a dish of the same sort. It serves as a comprehensive blueprint for culinary creation, encompassing not only the list of ingredients and their quantities but also the precise steps, techniques, and timing required to transform those ingredients into the desired dish [29]. In this study, only the recipe's essential building blocks, its ingredients, were taken into account, meaning cooking instructions and quantities were deliberately excluded from consideration.

Let I denote the set of all possible ingredients, where each ingredient is a distinct object. A recipe can then be represented as a subset of I, i.e., a collection of specific ingredients chosen from the set I, such that $R \subseteq I$.

2.2.1 Food Recognition

Food recognition is a critical task within the field of computer vision, with the goal of automatically identifying and categorizing food items using visual cues extracted from images. Recent advancements in deep learning, specifically convolutional neural networks (CNNs), have significantly enhanced the accuracy of food recognition systems. These models extract high-level features from food images, enabling precise identification of dishes and ingredients.

In [30], Salvador et al. introduced an inverse cooking system that utilizes food images to recreate cooking recipes. The system employs a series of steps, including image feature extraction, ingredient prediction, and encoding into ingredient embeddings. Subsequently, a cooking instruction decoder generates a recipe title and a sequence of cooking steps by considering image embeddings, ingredient embeddings, and previously predicted words. Through user studies, it was determined that the model outperformed the human baseline. Additionally, Chen et al. present two methods for ingredient recognition based on deep convolutional neural networks (DCNNs) [31].

Nonaka et al. [32] utilized a multimodal coupled network to evaluate cooking recipes by analyzing both images of cooked food and textual recipe descriptions. The aim of the study is to propose user-friendly recipes for viewers of cooking websites. The proposed method combines image and text data to estimate cooking recipes that are easily accessible and provide optimal conditions for users seeking recipe references.

Sugiyama et al. [33] proposed a method called Recipe Disentangled Embedding Generative Adversarial Network (RDE-GAN) to separate food image information into a recipe image feature and a non-recipe shape feature for encoding recipe images. Through the introduction of feature disentanglement, this model achieved superior performance compared to existing baselines in image-to-recipe and recipe-to-image retrieval tasks. Similarly, Xie et al. [34] employed a three-tier modality alignment approach to learn a joint embedding of text and image data. The focus of this model was to enhance the alignment between textual embeddings and the visual features of images, leading to comparable results.

More recently, to generate coherent and diverse recipes, a Decomposition Generation Network (DGN) with structure prediction is proposed by Wang et al. [35]. This approach divides the recipe generation process into structure prediction and content generation steps. By separating these processes, the model generates diverse recipes while maintaining coherence. Training involves a large-scale recipe dataset where food images and the corresponding ingredients are the model inputs.

2.2.2 Recipe Recommendation Systems

Most of the current models leveraging recipe data aim to recommend recipes to users by utilizing previous ratings of recipes provided by other users. These models consider a diverse range of users' ratings and feedback, enabling them to capture the intricate and personalized nature of culinary preferences. Advanced machine learning and recommendation algorithms are employed to identify patterns and correlations within the data, facilitating the generation of accurate and personalized recipe recommendations for individual users. As artificial intelligence-based tools, recommender systems have a primary objective of recommending recipes to users based on previous ratings given by themselves and other users. These systems play a crucial role in assisting online users by helping them navigate the overwhelming search space of available options and providing personalized information that aligns with their specific preferences and needs [36, 37].

The development of recommender systems has been driven by three main paradigms: content-based recommendation, collaborative filtering based recommendation, and hybrid recommendation. Content-based recommendations revolve around user profiling, item profiling, and profile matching, focusing on suggesting items that are similar to those previously consumed or preferred by the same users. On the other hand, collaborative filtering concentrates on leveraging the past preferences of similar users to generate recommendations for the active user. Lastly, hybrid recommender systems combine both content-based and collaborative filtering based recommendations [37].

In recent years, several research studies have proposed computational models for personalized food recommendation that integrate nutritional knowledge and user data. For instance, Toledo et al. [38] introduced a food recommender system capable of generating daily personalized meal plans for users based on their nutritional requirements and past food preferences. A study conducted by Chavan et al. [39] developed three recommender systems, each representing a different paradigm, incorporating individual calorie intake requirements. The hybrid recommender system demonstrated the best performance among the models.

In a more recent work, Rostami et al. [40] presented a novel health-aware food recommendation system that explicitly considers food ingredients, food categories, and the temporal aspect. This system employs time-aware collaborative filtering and a food ingredient content-based model to predict users' preferences.

2.2.3 Recipe Ideation and Food Pairing

Recipe ideation encompasses the creative process of generating new and innovative ideas for recipes. It serves as the foundation for culinary innovation, with food pairing, which consists in identifying ingredients or food items that harmoniously combine flavors and textures, being a fundamental aspect of this ideation process.

The study conducted by Ahn et al. [14] introduced the concept of food pairing and explored the relationship between food ingredients and their flavor profiles. Through an extensive analysis of a vast recipe database, the researchers discovered a pattern wherein ingredients sharing flavor compounds exhibited a tendency to be paired together, specially in western culinary traditions. This observation suggests that the selection of ingredient combinations in different cuisines may be guided by statistical co-occurrences or chemical similarities of ingredient pairs.

In Park et al. [41], KitcheNette, a model that utilizes Siamese neural networks, a type of deep learning architecture, to model the intricate connections between ingredients based on flavor profiles or other relevant characteristics, is introduced. The primary objective of the study was to develop a reliable and accurate model capable of predicting food ingredient pairing scores and offering valuable insights into the most suitable ingredient pairings. By training the network on a large dataset of food recipes, the researchers were able to recommend complementary food pairings and discover novel ingredient pairings. Subsequently, the authors of the aforementioned study introduced FlavorGraph, a comprehensive large-scale food graph [42]. FlavorGraph serves the purpose of generating representations of food and providing recommendations for food pairings. Furthermore, this extensive graph can also be used to predict the relationships between compounds and foods.

The process of generating personalized recipes refers to the development and creation of recipes tailored to individual preferences, dietary needs, and restrictions. This concept has been studied in the field of food science, which focuses on understanding the scientific aspects of food production, processing, and consumption.

In [43], Majumder et al. proposed a method to generate personalized recipes based on incomplete input specifications and user histories. This model utilizes an encoder-decoder framework, where the user provides input such as the dish name, a selection of a few key ingredients, and a specified calorie level. By leveraging this information, the model generates a recipe that is tailored to the user's taste preferences. The model demonstrated the ability to produce credible, customized, and logically coherent recipes that were favorably received by human evaluators for potential consumption.

Recipe generation was also explored by Lee et al. [44], where RecipeGPT, an online pre-trained transformer-based application, was presented. The research aimed to achieve two primary objectives: generating the ingredient list based on the recipe title and cooking instructions, and generating the cooking instructions given the recipe title and the list of ingredients. To generate the recipes, the system architecture relied on GPT-2, a generative pre-trained transformer model. The model incorporated an evaluation system to identify overlapping ingredients and compare the generated recipes to reference recipes. The performance of the system was evaluated using the F1 score, which consistently yielded values between 0.7 and 0.8 for both the validation and test sets, demonstrating the system's reliability.

The recipe-generation model developed by Fujita et al. [45] used an encoder-decoder framework. This model aimed to generate personalized recipes by incorporating reinforcement learning and coverage loss techniques. By analyzing user preferences and past recipe information, the model successfully matched generated recipes to individual tastes. The encoder component extracted key representations from input ingredients and names, guiding the decoder in recipe generation. Comparative evaluation of four models revealed that the proposed model outperformed the others.

2.2.4 Recipe Completion

Recipe completion is a task that can be executed by various methods, including natural language processing (NLP) models, where the model generates complete recipes based on partial or incomplete input. These models employ mostly machine learning techniques to predict the missing components of a recipe when provided with a partial description or list of ingredients by learning the patterns and structures commonly found in recipes. Drawing upon the input and learned patterns, the model generates the missing elements, which may include additional ingredients, measurements, cooking times, or detailed step-by-step instructions. Recipe completion models rely on statistical patterns to make predictions, leveraging the acquired knowledge to generate coherent and plausible recipes.

RecipeBowl, developed by Gim et al. [46], is a model that consists of a set encoder and a 2-way decoder, designed for predictive tasks in the culinary domain. By taking a set of ingredients and cooking tags as input, RecipeBowl offers suggestions for possible ingredient and recipe choices. The model predicts ingredients that were previously removed from the original recipe, enabling its completion and/or augmentation.

Gim et al. [47] later introduced RecipeMind, a model that predicts food affinity scores, which measure the suitability of including an ingredient in a set of other ingredients. This model utilizes co-occurrence statistics to determine the affinity scores. Furthermore, the paper proposes the use of a cascaded set transformer, enabling the joint learning of features between the current ingredient set and its additional ingredient.

Cueto et al. [48] investigates the utilization of item-based collaborative filtering to suggest missing ingredients for incomplete or partial recipes. The study focuses on addressing the challenges of sparse, high-dimensional data and aims to enhance the recommendation accuracy by leveraging co-occurrence patterns between ingredients.

In [49], De Clercq et al. analyzed the ability of different models to complete recipes with intentionally removed ingredients. The models evaluated were non-negative matrix factorization (NMF) and two-step Regularized Least Squares (RLS). NMF can only consider one dataset and requires a linear regression step for predicting new recipes. The RLS performs two regressions and can suggest new ingredients not yet present in any existing recipes, a capability that NMF lacks. Among the evaluated methods, the adapted two-step recursive least squares exhibited superior performance in terms of results.

This work, within the framework of recipe completion, leverages computational methods, as explored in the following section, to optimize the synthesis and refinement of recipes, integrating algorithmic strategies for enhanced and automated recipe completion and generation.

2.3 Prediction Models and Recommendation Systems

A recommendation system is an advanced software tool that utilizes data mining, machine learning algorithms, and historical user preferences to deliver personalized item suggestions, thereby facilitating users in their decision-making processes [50].

One of the key techniques employed in recommendation systems is Collaborative Filtering (CF), which plays a pivotal role in enhancing the quality of recommendations. This concept was initially introduced in 1992 by Goldberg et al. [51], who developed Tapestry, an experimental mail system designed to address the issue of mail overload and assist users in filtering documents relevant to their interests.

CF stands out as a recommendation technique that predicts a user's item preference by leveraging the historical behavior and choices of users who share similar patterns. The fundamental concept behind CF-based algorithms is rooted in the assumption that users who have agreed on certain items in the past are likely to agree on other items in the future. As a result, collaborative filtering methods generate personalized recommendations for users solely based on patterns of ratings or usage, without requiring external information about the items or users. Essentially, these algorithms offer item suggestions or predictions derived from the preferences of other users who have similar tastes and opinions. This method is commonly used in e-commerce websites, to propose personalized products, in streaming services, to recommend movies or music, or in news websites [50, 52].

Mathematically, collaborative filtering involves constructing a user-item matrix where each cell represents a user's rating or preference for an item. By analyzing this matrix, the CF algorithm used identifies similarities between users or items [50].

When it comes to measuring similarities in collaborative filtering, there are two distinct approaches. First, user-based CF identifies users with similar preferences and recommends items based on what those similar users have liked in the past. On the other hand, item-based CF focuses on item similarity, suggesting items that are related to the ones the target user has shown interest in. Both these approaches effectively utilize historical behavior and preferences to generate personalized recommendations, although they differ in the way they measure similarity and present item suggestions to users [50, 53].

However, the majority of the current collaborative filtering methods are not capable of effectively dealing with very large datasets. To overcome these challenges, one can leverage deep learning models like the restricted Boltzmann machine, specifically designed for collaborative filtering, since these have the ability to find hidden patterns in an unsupervised manner [54, 55].

2.3.1 Stochastic Processes

A stochastic process is a mathematical model that describes the evolution of a system over time in a probabilistic manner, i.e. a sequence $X_1, X_2, \ldots$ of random elements of some set is called a stochastic process [56]. Markov chains, named after the mathematician Andrei Markov, are a specific class of stochastic processes. These mathematical models, used to study systems that undergo transitions between different states over discrete time steps, are based on Markov's writings [57] in the early 20^{th} century [58].

Furthermore, a stochastic process can be considered a Markov chain if it exhibits a distinctive property called the Markov property [59]. The Markov property states that the future behavior of a system depends only on its current state and is independent of its past history. This means that a sequence $X_1, X_2, ...$ of random elements is a Markov chain if the conditional distribution of X_{n+1} depends on X_n only [56]. This simplifies the analysis of the system by focusing on the current state and the probabilities of transitioning to future states, without needing to consider the entire history of the system. For a process with discrete time, the Markov property can be defined by Eq.2.1 [59].

$$P(X_{n+1} = x_{n+1}|X_n = x_n, X_{n-1} = x_{n-1}, ..., X_1 = x_1) = P(X_{n+1} = x_{n+1}|X_n = x_n) \tag{2.1}$$

This property implies that the system's behavior can be fully characterized by the probabilities of transitioning between different states.

Let I be a countable set, where each $i \in I$ is referred to as a state, and I itself is known as the state space. In the context of Markov chains, the behavior of a system is described by transitioning between these states over time. The transition matrix, denoted as π, is a key component of Markov chains, and it characterizes the probabilities of transitioning between different states [60].

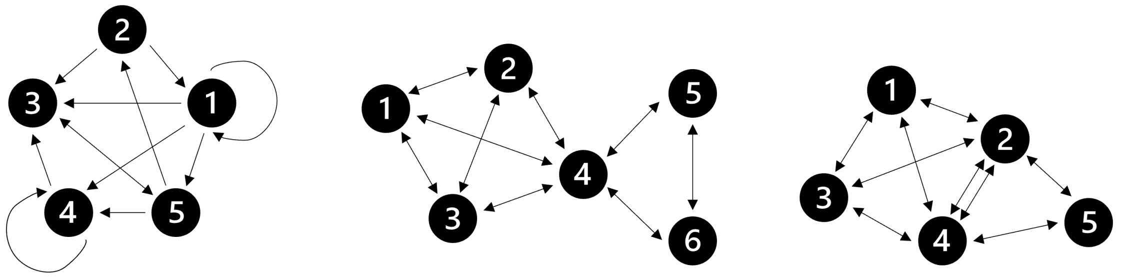
A measure on I is defined as $\lambda = (\lambda_i : i \in I)$, where $0 \leq \lambda_i < \infty$. If the sum of all elements in λ equals 1, i.e., $\sum_{i \in I} \lambda_i = 1$, then λ is referred to as a distribution. This distribution provides a probabilistic representation of the likelihood associated with each state [60, 61].

In the context of Markov chains, the transition matrix π is considered stochastic if each column $(\pi_{ij} : i \in I)$ of π forms a distribution. These transition probabilities π_{ij} represent the conditional probability of transitioning to state x_j at time $n + 1$, given that the current state is x_i at time n (Eq.2.2).

$$P(X_{n+1} = x_i|X_n = x_j) = \pi_{ij} \tag{2.2}$$

where $\sum_j \pi_{ij} = 1$.

Furthermore, the transition probabilities π_{ij} for each state i are equal to the corresponding element λ_j of the distribution associated with that state, i.e. $\pi_{ij} = \lambda_j$. The condition $\sum_{j \in I} \lambda_j = 1$ signifies that the elements of each row sum to 1, ensuring that the system moves to a new state at each time step [60, 56]. In Fig.2.2 there are represented three small Markov chains as an example, as well as their respective transition matrices.



$$\begin{pmatrix} 1/4 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/3 \\ 1/4 & 1/2 & 0 & 1/2 & 1/3 \\ 1/4 & 0 & 0 & 1/2 & 1/3 \\ 1/4 & 0 & 1 & 0 & 0 \end{pmatrix}$$

a)

$$\begin{pmatrix} 0 & 1/3 & 1/3 & 1/5 & 0 & 0 \\ 1/3 & 0 & 1/3 & 1/5 & 0 & 0 \\ 1/3 & 1/3 & 0 & 1/5 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/5 & 0 & 1/2 \\ 0 & 0 & 0 & 1/5 & 1/2 & 0 \end{pmatrix}$$

b)

$$\begin{pmatrix} 0 & 1/5 & 1/3 & 1/5 & 0 \\ 1/3 & 0 & 1/3 & 2/5 & 1/2 \\ 1/3 & 1/5 & 0 & 1/5 & 0 \\ 1/3 & 2/5 & 1/3 & 0 & 1/2 \\ 0 & 1/5 & 0 & 1/5 & 0 \end{pmatrix}$$

c)

Figure 2.2: Three examples of Markov chains and their respective transition matrices.

For example, Fig.2.2.b) and Fig.2.2.c) can be seen as Markov chains symbolizing different sets of recipes. Fig.2.2.b) depicts a set comprising two recipes. The first recipe encompasses ingredients $1, 2, 3$ and 4, while the second recipe includes ingredients $4, 5$ and 6. In contrast, Fig.2.2.c) represents another recipe set consisting of two recipes. The first recipe consists of ingredients $1, 2, 3$ and 4, whereas the second recipe encompasses ingredients $2, 4$ and 5.

Concerning instances cited in literature, Google PageRank, a foundational algorithm that has played a pivotal role in revolutionizing web search and ranking, stands as one of the most renowned applications of Markov chains [62]. Introduced by Larry Page and Sergey Brin, the founders of Google, in the late 1990s, PageRank aimed to address the challenge of efficiently organizing and ranking web pages on the internet [63, 64]. This algorithm laid the foundation for Google's rapid rise to dominance as a search engine and is still a crucial component of their search methodology. The first version of Google's search engine was introduced in 1998, prominently featuring the PageRank algorithm as a core component [65].

Before its introduction, most search engines relied heavily on simplistic keyword matching, which often resulted in less relevant search results [66]. In contrast, PageRank sought to evaluate the quality and importance of web pages based on their inbound and outbound links. Considering that the Word Wide Web (WWW) can be perceived as a directed graph, each web page is a state, or node, in the Markov chain and the transitions between nodes (web pages) are determined by the network of hyperlinks interconnecting them [62, 67].

PageRank's fundamental principles become evident through its operation. Initially, each web page is granted an identical PageRank score, usually shared uniformly among all pages (Eq.2.3).

$$P_r(p, t = 0) = \frac{1}{N} \tag{2.3}$$

for every web page p, where $P_r(p, t = 0)$ is the PageRank of page p in the initial step and N is the total number of web pages.

The PageRank score of a web page is then distributed among its outbound links (Eq.2.4). Web pages with more outbound links distribute their PageRank value more thinly among them. Furthermore, a web page's PageRank is influenced by the PageRank scores of the pages that link to it. Pages with higher PageRank scores that link to a particular page transfer a more substantial PageRank value. This iterative refinement process continues until PageRank scores reach a state of equilibrium, effectively capturing the relative importance of web pages [62, 67].

$$P_r(p, t + 1) = \sum_{q \in M(p)} \frac{P_r(q, t)}{N(q)} \tag{2.4}$$

where $M(p)$ is the set of web pages that link to page p and $N(q)$ is the number of outbound links from page q.

Furthermore, Markov processes find application in predicting user choices within recommendation systems. These models sequentially capture state changes, assuming that future states depend on the present state rather than past ones [68]. Markov chain models enable direct state observation, and in the context of recommendation systems, they forecast users' next selections by optimizing over sequential criteria. Such models are referred to as sequence-aware or time-aware RS, utilizing ordered or timestamped records of past interactions [69, 70].

Khorasani et al. [71] introduced a Markov-based collaborative filtering methodology for recommending University courses, focusing on guiding students in their course selections each semester based on their prior course history. Moreover, student course sequences were treated as stochastic processes, enabling the estimation of transition probabilities from empirical data. Two distinct estimation approaches were employed: basic Maximum Likelihood Estimation (MLE) and an enhanced MLE technique employing skip-gram modeling. Empirical evidence highlighted the significant influence of course order on academic outcomes. Within this framework, the skip Markov model demonstrated a Recall rate of 78% and a Precision of 23% when tested on a dataset containing ten years of past enrollment information (2001-2011) sourced from a Canadian research university.

Aghdam et al. [72] introduce an approach to context-aware recommender systems using a hierarchical hidden Markov model. The method effectively captures changes in user preferences over time by modeling latent user contexts. By incorporating the current user context as a hidden variable, the proposed model leverages feedback sequences to automatically learn user-specific latent contexts. Experimental results on benchmark datasets demonstrate the superior performance of this model in comparison to other methods. The paper addresses the challenge of adapting recommendations to evolving user preferences and contextual changes, showcasing the potential of utilizing predicted contexts to enhance recommendation diversity.

2.3.2 Restricted Boltzmann Machines

The restricted Boltzmann machine is a probabilistic model, first introduced by Paul Smolensky in 1986 [73], as a two-layer probabilistic neural network capable of learning a probability distribution over its input as a generative model. Even though it was originally developed in the context of cognitive psychology, due to the increase in computational power and the development of faster learning algorithms, the RBM was later rediscovered in the machine learning field as an unsupervised learning algorithm capable of extracting meaningful features from unlabeled data, such as images or text. The popularity of RBMs grew significantly in the 2000s, especially in the field of deep learning, where it was used as a building block for more complex neural networks, such as deep belief networks and deep autoencoders [73, 74, 75, 76].

Currently, RBMs are used in a variety of applications, including problems with high dimensional data, such as image recognition [77], collaborative filtering [55], and speech recognition [78] and, due to its strong capability of representing dependency in data, being used as a basic building block for other deep learning models [74, 75, 79].

The RBM is, therefore, a type of artificial neural network of stochastic units with undirected connections between pairs of units in two layers: a visible layer $\vec{v}$ and a hidden layer $\vec{h}$, which are usually binary. The visible layer represents the input data and the hidden layer represents the learned features of the input data. The goal of an RBM is to learn a probability distribution over the input data by adjusting the weights between the visible and hidden layers [74, 75].

Moreover, the RBM is a generative stochastic network with m elements in the visible unit $\vec{v}$ and n elements in the hidden unit $\vec{h}$, as represented in Fig.2.3. The graph of an RBM depicts connections solely between the hidden and visible variables, while no connections exist between two variables of the same layer. From a probabilistic perspective, this indicates that given the visible variables' state, the hidden variables are independent, and vice versa [76].

The probability distribution of an RBM can be modeled using an energy function when both the visible and hidden units are binary. The energy of a joint configuration (v, h) of the visible and hidden units can be computed using Eq.2.5 [80].

$$E(v,h;\theta) = -\sum_{i=1}^{m}\sum_{j=1}^{n} v_i W_{ij} h_j - \sum_{j=1}^{n} c_j h_j - \sum_{i=1}^{m} b_i v_i \tag{2.5}$$

where $v_i \in \{0,1\}$ and $h_j \in \{0,1\}$ are the binary states of the visible unit i and the hidden unit j and $\theta = (W, b, c)$ are the parameters that need to be determined. $W = (W_{ij}) \in R^{n\times m}$ represents the weight connecting the i^{th} visible unit and the j^{th}

hidden unit and $b = \{b_i\}_{i=1}^{m}$ and $c = \{c_j\}_{j=1}^{n}$ are the bias terms of the visible and the hidden layers, respectively. These parameters are determined through the process of training, wherein techniques such as Contrastive Divergence are employed to iteratively adjust the values of W, b, and c, as will be elaborated in Chapter 3 [76, 79, 80].

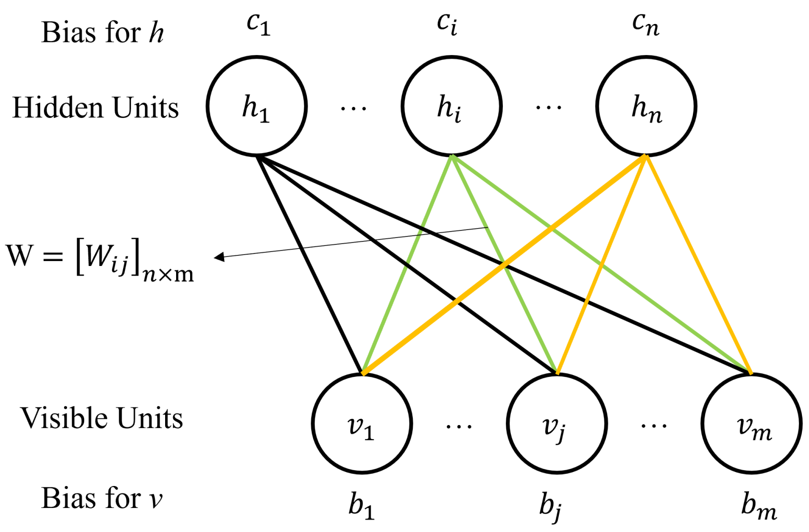


Figure 2.3: RBM with m visible units and n hidden units. W_{ij} is the weight between h_i and v_j and the terms b and c denote the bias for visible and hidden unit, respectively.

Furthermore, the probability of a configuration (v, h) can be computed using the energy function. The joint probability distribution of the RBM is given by the Gibbs distribution, represented in Eq.2.6 [76].

$$P(v, h; \theta) = \frac{1}{Z(\theta)} \mathrm{e}^{-E(v,h;\theta)} \tag{2.6}$$

where $Z(\theta)$ is the partition function, defined as the sum of the exponential of the energy function over all possible configurations of (v, h) (Eq.2.7) [79].

$$Z(\theta) = \sum_{v}\sum_{h} \mathrm{e}^{-E(v,h;\theta)} \tag{2.7}$$

Moreover, from Eq.2.6, it is possible to calculate the probability that the network assigns to a visible vector, v. This is the marginal distribution, which refers to the probability distribution of a subset of variables. In this case, the marginal distribution of the visible variables refers to the probability distribution of all possible configurations of the visible units, without taking into account the values of the hidden units, and is given by summing over all possible hidden vectors, as represented in Eq.2.8 [76, 80].

$$P(v; \theta) = \frac{1}{Z(\theta)} \sum_{h} \mathrm{e}^{-E(v,h;\theta)} = \frac{1}{Z(\theta)} \prod_{j=1}^{m} \mathrm{e}^{b_j v_j} \prod_{i=1}^{n} (1 + \mathrm{e}^{c_i + \sum_{j=1}^{m} W_{ij} v_j}) \tag{2.8}$$

The partition function ensures that Eq.2.8 is a valid probability distribution, i.e. that the probabilities sum up to 1 over all possible states of the visible and hidden units [75].

In the context of recommender systems, Salakhutdinov's work [55], which was one of the pioneering uses of RBMs for collaborative filtering, employed an RBM to model user-item interactions and generate recommendations using a Netflix dataset containing 100 million user/movie ratings. The approach involved creating an RBM for each user, with the input matrix V being a binary matrix of size $k \times m$, where m represents the number of movies the user watched, and k represents the number of different possible ratings. Each entry $v_{i,j}$ in the matrix is assigned a value of 1 if the user gave rating i to movie j, and 0 otherwise. This method demonstrated remarkable performance as a top-tier collaborative filtering model, surpassing Netflix's own rating system by over 6%.

Behera et al. [81] introduced a recommendation model that utilizes the RBM to predict users' preferences for movies they have not watched. The RBM effectively handled sparse datasets, enabling accurate estimation of missing user ratings and providing personalized movie recommendations. Compared to traditional methods, like Pearson correlation and average rating prediction, the RBM-based model outperformed in terms of mean absolute error (MAE), calculated by subtracting the recommendation score from the actual rating and divided by the number of movies considered, showcasing its ability to generate more precise movie recommendations.

2.3.3 Non-Negative Matrix Factorization

Matrix factorization is a fundamental mathematical technique in the field of machine learning and data analysis. Its primary objective is to decompose a given matrix into two or more lower-dimensional matrices while preserving the matrix's essential properties. This process, by maintaining the matrix's fundamental characteristics, allows the model to discern hidden patterns or latent factors [82, 83].

One of the earliest matrix factorization models is the Singular Value Decomposition (SVD), which decomposes a matrix into three constituent matrices: U, Σ, and V^T. Remarkably, SVD's development dates back to the 19th century, with several mathematicians independently contributing to its discovery [84].

While SVD laid the foundation for matrix factorization, another significant advancement in matrix factorization models came in the form of non-negative matrix factorization, introduced in 1994 [85]. NMF is an unsupervised mathematical technique extensively employed in data analysis and machine learning. It plays a pivotal role in decomposing complex data into simpler components while ensuring that all values in the resulting matrices are non-negative. This property promotes interpretability and enhances the extraction of meaningful data representations, aligning with problems where only positive numbers are acceptable, such as situations associated with probabilities, where negative numbers do not have mathematical meaning [83, 86, 87].

NMF finds applications across various domains, including natural language processing, image analysis, and recommendation systems. Its primary goal is to automatically extract hidden patterns from high-dimensional data vectors. With its interpretability and versatility, NMF remains a fundamental technique for uncovering meaningful insights from complex datasets and it is commonly used for dimensional reduction, unsupervised learning, and prediction [82, 87].

The NMF algorithm operates by starting with a matrix denoted as V, containing only non-negative values, representing the original input data. The objective is to discover two non-negative matrices, W and H, of significantly lower rank. These matrices are chosen in such a way that their linear combination, represented by Eq. 2.9, closely approximates the original matrix V, as represented in Fig.2.4.

$$V_{(m\times n)} \approx R(W,H) = W_{(m\times k)}H_{(k\times n)} \tag{2.9}$$

where $k << \min(m,n)$.

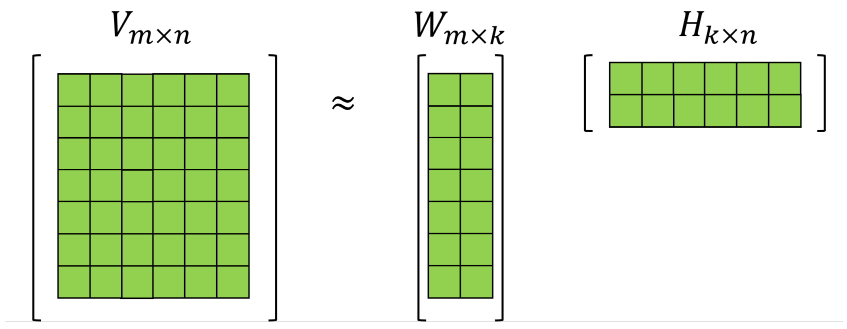


Figure 2.4: Schematic representation of the NMF algorithm.

The core objective of NMF is to factorize matrix V into matrices W and H while minimizing an objective function that quantifies the disparity between V and the estimated rank matrix R. This objective function, also known as a divergence measure, can take various forms, such as Euclidean distance or Kullback–Leibler (KL) divergence. These divergence measures are used to assess how well the factorization aligns with the original data [82, 83].

The Euclidean distance, also known as the Frobenius norm, was the objective function used in this study. The objective is to minimize the sum of squared differences between the original data matrix V and the product of the factorized matrices WH. This is formally expressed by Eq.2.10.

$$\min_{W,H} f(W,H) = \frac{1}{2}\left\|V - WH\right\|_F^2, \; s.t.\, W,H \geq 0 \tag{2.10}$$

where $\|\cdot\|_F$ represents the Frobenius norm, measuring the element-wise difference between matrices [82, 83, 88].

Matrix factorization algorithms, NMF and others, are commonly used in recommendation systems for product recommendations. This is made by constructing a user-item matrix, where each row is an user and each column represents an item. Then, by reconstructing the original matrix, calculating WH, the model gives a prediction of what the users would rate the items they have not interacted with [82, 89].

Khan et al. [90] proposed a novel recommender system built upon a semantics-based item content embedding model, enriched with contextual features extracted through CNN. They incorporated NMF as a collaborative filtering technique, enhanced with improvised embedding, and termed it Contx-NMF. This hybrid collaborative filtering technique aims to predict ratings in situations with sparse user-to-item ratings. To evaluate the performance of their model, the researchers conducted experiments using three well-known public datasets: MovieLens 1M, MovieLens 10M, and Amazon Instant Video (AIV), having achieved better precision, recall, accuracy and area under curve than other state of the art techniques.

In the area of transfer learning, a technique in machine learning in which knowledge learned from a task is re-used in order to boost performance on a related task, Ievgen and Younés [91] proposed an unsupervised transfer learning approach that seeks to find a partition of unlabeled data in a target domain by utilizing knowledge obtained from clustering unlabeled data in a source domain. The methodology involves identifying partitions in various feature subspaces of a source task to improve the accuracy of the partition in the target domain. From the set of source partitions, the researchers select k nearest neighbors based on a similarity measure. Finally, they apply multi-layer non-negative matrix factorization to derive a partition of objects in the target domain.

Chapter 3

Methods and Techniques to Test Performance

In this chapter, the methodology employed for both recipe completion and the assessment of how effectively these methods encapsulate the underlying patterns within a recipe database are presented. The subsequent sections contain a comprehensive analysis of the techniques utilized in this study: Markov and non-Markovian chains, restricted Boltzmann machines, and non-negative matrix factorization. These methods are evaluated using mutual information and correlation coefficients between ingredient pairs, and hierarchical clustering is used to find groups of ingredients commonly used together, which will be explored in the last sections.

3.1 Stochastic Processes

In the culinary context, each stage of a recipe introduces stochastic elements, variables influenced by ingredient choices. The resultant dish reflects a probabilistic interplay of these factors, resembling the unpredictable state transitions observed in stochastic processes.

3.1.1 Markov Chains

A recipe can be viewed through the lens of a Markov chain, where each state corresponds to an ingredient. In this particular context, the interconnections between states operate bidirectionally. Consequently, if the conditional probability $P(i|j)$ is not equal to zero, it follows that $P(j|i)$ is also not equal to zero. Conversely, if $P(i|j)$ equals zero, then $P(j|i)$ will also equal zero.

To establish the connections between ingredients, the sets representing recipes have to be examined. If a given set contains both elements i and j, then these two states are considered connected. The strength of this connection is determined by the number of recipes that include both ingredients. Specifically, if there are n recipes containing both ingredients i and j, then these states possess an n-tuple connection. Additionally, Fig.2.2.b) and Fig.2.2.c) can be interpreted as Markov chains that represent distinct recipe collections, as explained before.

Therefore, Markov chains can be a valuable tool for recipe completion. The process involves creating a transition matrix based on an initial dataset of recipes, as explained below. This matrix serves as the foundation for suggesting the subsequent ingredients required to achieve a complete recipe.

Through the assessment of the co-occurrence of elements i and j, commonly known as the support of the pair (i, j) and represented as $s(i, j)$, it becomes possible to construct a symmetric matrix denoted as π_0. This matrix exhibits zeros on the diagonal, while specific elements outside the diagonal may be zero as well. An estimation of the conditional probability of element i being present given that element j was added, as represented in Eq.3.1, defines a transition matrix π for a non-symmetric Markov chain.

$$P(j \to i) = P(i|j) = \frac{s(i,j)}{\sum_k s(k,j)} \tag{3.1}$$

The support of a pair $s(i, j) = s(j, i)$ represents in how many of the initial systems, used to construct π_0, elements i and j are both present. This means that $P(j \to i)$, which corresponds to the position π_{ij} in the transition matrix, is calculated by dividing the number of initial systems where elements i and j co-occur by the sum of the j^{th} column of matrix π_0. The transition from one state to another can be understood as the introduction of an element into a system, where the same element may be added multiple times.

This Markov chain can exhibit ergodicity (irreducibility) or non-ergodicity. In the case of ergodicity, starting from any state, it is possible to construct a system with a sufficiently large number of states that encompasses any of the possible elements [92]. If the Markov chain is not ergodic, the generated sets will be limited to a subset of the possible states, depending on the initial one.

In the case of an ergodic Markov chain, a unique stationary distribution emerges, which characterizes the likelihood of a specific element occurring within the generated sets. Let π_{ij} denote $P(j \to i)$, the transition probability from element j to element i. Consequently, the recursive relation $P_t = \pi P_{t-1}$ holds, where P_t signifies the probability distribution at time t (as described by Eq. 3.2).

$$P_{t,i} = \sum_k \pi_{ik} P_{t-1,k} \tag{3.2}$$

where $P_{t,i}$ represents the probability of adding element i to a system after $t - 1$ state transitions. Eq.3.2 signifies that the probability of adding element i at time t is determined by summing over all possible preceding states k, with π_{ik} denoting the transition probability from state k to state i. By iteratively updating the probability distribution over time, it is possible to gain insight into the long-term behavior of the Markov chain.

In the long-term of the Markov chain, where the number of elements added to the system increases (including repetitions), the limit as t approaches infinity, $lim_{t\to\infty}P_t = P_\infty = lim_{t\to\infty}\pi^t P_0$, becomes independent of the initial systems, P_0, from which the Markov chain starts [92].

To initiate the ingredient suggestion process, a random ingredient (i) is chosen from the incomplete recipe, which needs n more ingredients. Then, another ingredient (j) is randomly selected from the complete ingredient list, based on the probabilities associated with the i^{th} column of the transition matrix. Following this, another ingredient (k) is chosen using probabilities from the j^{th} column of the transition matrix. This selection process continues until n ingredients are added. It is important to note that these ingredients can be repeated, and in some cases, it might not be possible to add n new ingredients, indicating a non-ergodic Markov chain. Hence, it is crucial to repeat this process multiple times for reliable results.

By following this approach, Markov chains provide a systematic means to suggest a sequence of a maximum of n ingredients that can be incorporated into an incomplete recipe. This method leverages the information captured in the transition matrix and offers a structured approach to recipe completion.

3.1.2 Non-Markovian Chains

Based on the Markov chain methodology described above to complete recipes, non-Markovian chain approaches can be developed.

This can be done by introducing the restriction that if a set already contains a certain element, this can not be added again, introducing non-Markovicity in the stochastic process by having a memory effect in the addition of new elements to the set. This can also be described as a "self-avoiding walk", which is a random walk that does not auto-intersect [93]. The conditional probability is defined by Eq.3.3.

$$P(j \to i) = P(i|j) = \frac{s(i,j)}{\sum_{k\notin R} s(k,j)} \tag{3.3}$$

On another non-Markovian approach, the states are not added in a sequential manner, based on the last one previously attached. In this case, the probability of adding element i to a set R of different states, can be defined defined by Eq.3.4, which can be useful to limit addition of elements not compatible with at least one of the currently present ones, but can highly limit the addition of uncommon elements.

$$P(R \to i) = \frac{\prod_{j\in R} s(i,j)}{\sum_{k\notin R}\prod_{j\in R} s(k,j)} \tag{3.4}$$

Another way to add the states in a non-sequential manner is represented in Eq.3.5.

This approach allows less common elements to more often have a non-zero probability of being added to set R, being less likely to have $P(R \to i) = 0$ for every $i \notin R$. However it does not limit as well the addition of elements that are not compatible with just one or two elements in R.

$$P(R \to i) = \frac{\sum_{j \in R} s(i,j)}{\sum_{k \notin R} \sum_{j \in R} s(k,j)} \tag{3.5}$$

This last approach was the one used in the present study. It is possible to approximate $P(R \to i)$ based on the transition matrix π, as following in Eq. 3.6.

$$P(R \to i) = \frac{\sum_{j \in R} \pi_{j,i}}{\sum_{k \notin R} \sum_{j \in R} \pi_{j,k}} \tag{3.6}$$

Algorithm 1 shows the process necessary to suggest new ingredients to an incomplete recipe. However, it is important to mention that to assure a significant statistical ensemble, the process needs to be repeated multiple times, as it is represented by the number of samples.

Algorithm 1 Non-Markovian chain

```
1: $P \leftarrow$ square transition matrix
2: $C \leftarrow$ list initialized as 0 for every ingredient
3: $n \leftarrow$ maximum number of ingredients to add
4: $R \leftarrow$ list of ingredients of an incomplete recipe
5: $ns \leftarrow$ number of samples
6: for sample in $ns$ do
7:     $ni \leftarrow$ initialize as 0
8:     while $ni < n$ do
9:         if $\sum_{k \notin R} \sum_{j \in R} P_{j,k} = 0$ then
10:            break
11:        end if
12:        $Pn \leftarrow \sum_{j \in R} P_{j,i} / \sum_{k \notin R} \sum_{j \in R} P_{j,k}$, for every $i \notin R$
13:        $r \leftarrow$ random ingredient selected according the probabilities of $Pn$
14:        $ni \leftarrow$ add 1
15:        $C \leftarrow$ add 1 to the $r^{th}$ position of the list
16:    end while
17: end for
18: $R \leftarrow$ add the $n$ ingredients with highest value in $C$
```

3.2 Restricted Boltzmann Machine

The RBM can be viewed as a stochastic neural network, where the nodes and edges correspond to neurons and synaptic connections, respectively. In this context, the bipartite structure of the RBM implies that the visible and hidden units are conditionally independent of each other. This means that the probability distribution of the visible units depends only on the state of the hidden units, and vice versa, as seen in Eq.3.7 and Eq.3.8. Furthermore, the conditional probability of a single variable being one can be seen as the firing rate of a (stochastic) neuron with a sigmoid activation function [74, 76].

$$P(v_i = 1|h) = \sigma(\sum_j W_{ij} h_j + c_i) \tag{3.7}$$

$$P(h_j = 1|v) = \sigma(\sum_i v_i W_{ij} + b_j) \tag{3.8}$$

where $\sigma(x) = 1/(1 + exp(-x))$ is the sigmoid function [79].

Computing $P(v, h; \theta)$ (Eq.2.6) or $P(v; \theta)$ (Eq.2.8) is usually unmanageable. However, since there is independence between the variables in one layer, Gibbs sampling can be used. Gibbs sampling is a Markov Chain Monte Carlo (MCMC) algorithm commonly used for generating samples from probability distributions that are difficult to sample directly. It iteratively samples each variable's conditional distribution while holding the other variables fixed. This process is repeated for all variables in the model, generating the next state x^{s+1} from the current state x^s, and the resulting sequence of samples converges to the true distribution of interest [74, 75, 94].

Moreover, all of the hidden units can be sampled jointly, since they are conditionally independent given the visible units, and vice-versa for the visible units. This sampling method, which is called block Gibbs sampling, updates many variables simultaneously by allowing the Gibbs sampling to be performed in two steps: sampling a new state v for the visible layer based on $P(v|h)$ (Eq.3.7) and sampling a state h for the hidden neurons based on $P(h|v)$ (Eq.3.8) [74, 76].

To train an RBM, various algorithms can be used, but all common training algorithms for RBMs are designed to approximate the log-likelihood gradient based on some given data. The log-likelihood gradient is a mathematical way of measuring the difference between the predicted output of the RBM and the actual output. By approximating this gradient and performing gradient ascent on these approximations, the RBM can learn to adjust its weights and biases to better fit the data [76].

For the model represented in Eq.2.8 with parameters θ, the log-likelihood given a single training example v can be calculated by Eq.3.9. A standard way of estimating the parameters of a statistical model is through maximum-likelihood estimation. Therefore, RBM training corresponds to finding the parameters θ that maximize the likelihood given the training data [76].

$$\ln\mathcal{L}(\theta|v) = \ln P(v|\theta) = \ln\frac{1}{Z(\theta)}\sum_h \mathrm{e}^{-E(v,h;\theta)} = \ln\sum_h \mathrm{e}^{-E(v,h;\theta)} - \ln\sum_{v,h} \mathrm{e}^{-E(v,h;\theta)} \tag{3.9}$$

The gradient of Eq.3.9 represents the difference between two expectations, as seen in Eq.3.10. Specifically, it is the difference between the expected values of the energy function under the model distribution ($\langle v_i h_j \rangle_{model}$) and the conditional distribution of the hidden variables, given the training example ($\langle v_i h_j \rangle_{data}$) [76, 80].

$$\frac{\partial \ln \mathcal{L}(\theta|v)}{\partial \theta} = -\sum_{h} P(h|v) \frac{\partial E(v,h;\theta)}{\partial \theta} + \sum_{v,h} P(v,h;\theta) \frac{\partial E(v,h;\theta)}{\partial \theta} \tag{3.10}$$

where $P(h|v) = P(v,h;\theta)/P(v;\theta)$, being $P(v,h;\theta)$ given by Eq.2.6 and $P(v;\theta)$ given by Eq.2.8.

While it is possible to approximate these expectations by drawing samples from the corresponding distributions using MCMC techniques, it usually necessitates numerous sampling steps to obtain unbiased estimates of the log-likelihood gradient, making the computational costs too large.

Therefore, the RBM is trained through contrastive divergence, which seeks to optimize the weights and biases to maximize the log-likelihood of the training data. This is accomplished by initializing the Markov chain with samples from the data distribution and subsequently running it for a number of steps to generate samples that approximate the model distribution. The estimated gradient of the log-likelihood, obtained using these samples, is then used to update the model parameters [76, 95].

The values of the hidden units in binary form are calculated simultaneously using Eq.3.8. After determining the binary states for the hidden units, a reconstruction is generated by setting each v_i to 1 with a probability determined by Eq.3.7. The update rule for the weight matrix is represented in Eq.3.11 [80].

$$\Delta W_{ij} = \alpha \left(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model} \right) \tag{3.11}$$

where $\langle v_i h_j \rangle_{data}$ is the expected value of the product of the i^{th} visible unit and the j^{th} hidden unit under the distribution of the training data, $\langle v_i h_j \rangle_{model}$ is the expected value of the product of the i^{th} visible unit and the j^{th} hidden unit under the model distribution and α is the learning rate [80]. This hyper-parameter determines the step size in parameter updates during training. A higher learning rate can lead to faster convergence but may result in overshooting the optimal solution, while a lower learning rate may lead to slower convergence but more stable training [96].

Moreover, the biases for the visible and hidden layers are updated using the rules in Eq.3.12 and Eq.3.13.

$$\Delta b_i = \alpha \left(\langle v_i \rangle_{data} - \langle v_i \rangle_{model} \right) \tag{3.12}$$

$$\Delta c_j = \alpha \left(\langle h_j \rangle_{data} - \langle h_j \rangle_{model} \right) \tag{3.13}$$

where $\langle v_i \rangle_{data}$ is the expected value of the i^{th} visible unit under the distribution of the training data, $\langle v_i \rangle_{model}$ is the expected value of the i^{th} visible unit under the model distribution, $\langle h_j \rangle_{data}$ is the expected value of the j^{th} hidden unit under the distribution of the training data, and $\langle h_j \rangle_{model}$ is the expected value of the j [80].

After the RBM parameters have been learned, the model can be used to generate new samples from the learned distribution. This is typically accomplished by performing Gibbs sampling for a fixed number of steps, denoted by k. The value of k determines the extent of mixing between the model distribution and the data distribution, and it is usually set to a small value, such as 1, the value used in this work, to ensure fast convergence. This process is known as k-step contrastive divergence learning (CD-k) [74, 76].

Regarding this study, the training of RBM was made employing TensorFlow, an open-source platform for machine learning and artificial intelligence. The RBM's computational process involves several essential steps, as seen in Algorithm 2.

First, the process starts with a dataset, which serves as the foundation for the RBM model, where each data point corresponds to a recipe. The input to the model consists of x neurons, also known as visible units. Here, x represents the total number of ingredients present in the dataset, ni. Each neuron assumes a binary state, either 0 or 1, denoting the absence or presence of a specific ingredient in a given recipe, respectively.

Then, during the training process, the dataset was divided into batches, with each epoch processing a fixed number of batches, since this enhances computational efficiency. In this case, there were used 10 batches, each containing 100 data points per epoch and the learning rate (α) was set to 0.1.

The CD can be seen as a matrix of values used to adjust the weight matrix and is updated incrementally over multiple training steps (epochs). The update equation for the weight matrix is expressed in Eq. 3.11, such that $CD = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$.

Finally, to evaluate the recommendation system's performance, the MAE metric is used to quantify the dissimilarity between the data and its reconstruction, providing a measure of the model's accuracy.

Upon completing the model's training, recommendations can be generated through the process of passing the visible units through the RBM's hidden layer. This step signifies the activation of the hidden layer based on the current state of the visible layer. Subsequently, the model undergoes a reconstruction phase where the probabilities of the visible units activating are calculated based on the current state of the hidden layer. This step aims to reconstruct the input data, utilizing the output derived from the hidden layer.

To calculate the visible layer probabilities, Eq.3.8 is applied, resulting in a list of probabilities that represent the likelihood of neuron activation. These probabilities are then normalized, ensuring that their combined sum equals 1.

Algorithm 2 RBM training algorithm

```
1: $Y \leftarrow$ Load binary input data
2: Initialize RBM parameters: $W$, visible bias $b$, and hidden bias $c$
3: $epochs \leftarrow$ Number of epochs
4: $errors \leftarrow$ Initialize error list
5: for $epoch$ in $epochs$ do
6:     for each data point in batch do
7:         Pass data point through network, calculating $v_i$ and $h_i$
8:         Calculate $\langle v_i h_j \rangle_{data}$ and $\langle v_i h_j \rangle_{model}$
9:         Update parameters: $W$ (Eq.3.11), $b$ (Eq.3.12), and $c$ (Eq.3.13)
10:    end for
11:    $errors \leftarrow$ append MAE to the list.
12: end for
```

In this specific context, the initial state of visible units represents an incomplete recipe. Accordingly, a new ingredient is selected for addition to the recipe. The choice of this ingredient is contingent upon the probabilities emanating from the normalized reconstructed visible layer.

3.3 Non-Negative Matrix Factorization

In the present study, the matrix denoted as V (see Eq.2.9) assumes the form of a binary matrix, wherein each row corresponds to a recipe, and each column corresponds to an ingredient.

Upon obtaining matrices W and H (such that $V \approx WH$), the primary objective is not the generation of novel ingredient recommendations for recipes within the existing database. Instead, the goal is to offer ingredient suggestions for entirely new users or, in this case, previously unseen recipes. This necessitates the computation of matrix X such that it satisfies the equation $VX = WH$, a task accomplished through Eq.3.14.

$$X = V^{+}WH \tag{3.14}$$

being $V^{+} = (V^{T}V)^{-1}V^{T}$ the pseudo-inverse of matrix V.

In conventional recommendation systems geared towards rating predictions, matrix $R = WH$ is employed. To predict the missing ratings assigned to a particular user, u, one typically computes R_u. However, an analogous outcome can be achieved by multiplying the row corresponding to user u by matrix X. This equivalence is expressed as $R_u = V_u X$.

Within the domain of recipe-based recommendations, the primary objective is to propose novel ingredients for incomplete recipes that have never been encountered previously. This is realized by performing a matrix-vector multiplication between the recipe vector r and matrix X, thereby yielding $r_{\text{predicted}} = rX$. In scenarios where the aim is to provide ingredient recommendations for multiple recipes concurrently, r may manifest as a matrix encompassing numerous incomplete recipes.

Lastly, Algorithm 3 summarizes the NMF-based methodology for ingredient recommendations to complete or enhance recipes.

Algorithm 3 NMF Algorithm

```
1: $V \leftarrow$ Load binary input data
2: $r \leftarrow$ list/matrix of ingredients of an incomplete recipe/recipes
3: Calculate optimized $W$ and $H$ based on the Frobenius norm
4: Calculate $X$ according to Eq.3.14
5: $r_{\text{predicted}} \leftarrow rX$
```

3.4 Mutual Information and Correlation Coefficient

Mutual information (I) is a fundamental concept in information theory that measures the amount of information shared between two random variables. It measures how much knowing the value of one variable reduces uncertainty about the other. For two discrete random variables X and Y, the mutual information is calculated using Eq.3.15 [97].

$$I(X;Y) = \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} P(x,y) \log\left(\frac{P(x,y)}{P(x)P(y)}\right) \tag{3.15}$$

where $I(X;Y)$ represents the mutual information between variables X and Y, $\mathcal{X}$ and $\mathcal{Y}$ are the sets of possible values of variables X and Y, respectively, $P(x,y)$ is the joint probability mass function of X and Y and $P(x)$ and $P(y)$ are the marginal probability mass functions of X and Y, respectively [97]. The logarithm is typically taken to base 2, resulting in mutual information being measured in bits [98].

Shannon entropy (H) is another important concept in information theory. Considering a random variable X existing within a discrete space $\mathcal{X}$, with x representing an element from $\mathcal{X}$, the entropy of the random variable X within the discrete space $\mathcal{X}$ serves as an indicator of the uncertainty or randomness of a single random variable and is calculated according to Eq.3.16. Since entropy quantifies the average level of unpredictability associated with the outcomes of a random variable, higher entropy indicates greater uncertainty [97].

$$H(X) = -\sum_{x\in\mathcal{X}} P(x) \log P(x) \tag{3.16}$$

where $P(x)$ is the probability mass function of X.

The relationship between mutual information and entropy is captured by Eq.3.17.

$$I(X;Y) = H(X) - H(X|Y) = H(X) + H(Y) - H(X,Y) \tag{3.17}$$

where $H(X|Y)$ is the conditional entropy of X given Y and $H(X,Y)$ is the joint entropy of X and Y, calculated according to Eq.3.18. The joint entropy captures the overall uncertainty associated with both variables considered together [97].

$$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(x,y)\log P(x,y) \tag{3.18}$$

Furthermore, Eq.3.17 highlights that mutual information is the reduction in uncertainty about X when Y is known, which is the difference between the initial uncertainty of X and the remaining uncertainty after Y is revealed, and vice-versa, since $I(X;Y) = I(Y;X)$.

Mutual information provides insight into how much knowing the value of one variable reduces uncertainty about the other. When X and Y are independent, their mutual information is zero, indicating that knowledge of one variable provides no information about the other. On the other hand, when mutual information is high, the variables are strongly dependent, and knowing the value of one variable significantly helps predict the value of the other [98].

Mutual information provides a powerful tool to quantify the information shared between random variables. Its relationship with entropy sheds light on how the uncertainty in one variable is affected by the knowledge of another. This understanding has broad applications in diverse domains, making mutual information a cornerstone of information theory.

In the context of recipes, ingredients can be viewed as binary variables: they are either present (1) or absent (0). Therefore, $\mathcal{X} = \{0,1\}$ represents the set of possible values for an ingredient's presence. For two distinct ingredients, denoted as i and j, the corresponding random binary variables are denoted as X and Y, taking values in $\mathcal{X}$. In this case, $\mathcal{X} = \mathcal{Y}$. Moreover, the probability that ingredient i is present in a recipe is given by Eq.3.19 and the joint probability of both ingredients i and j ($i \neq j$) being present is given by Eq.3.20.

$$P(X=1) = P(i) = \frac{\sum_{r=1}^{nr} S_r(i)}{nr} \tag{3.19}$$

where nr is the total number of recipes in the database and $S_r(i)$ is a binary indicator that equals 1 if recipe r contains ingredient i and 0 otherwise.

$$P(X=1, Y=1) = P(i,j) = \frac{\sum_{r=1}^{nr} S_r(i,j)}{nr} \tag{3.20}$$

where $S_r(i,j)$ equals 1 if recipe r contains both ingredients i and j, and 0 otherwise.

With these foundational probabilities established ($P(X = 1)$, $P(Y = 1)$ and $P(X = 1, Y = 1)$), it is possible to derive the others, as outlined in Tab.3.1.

Table 3.1: Conditional probability table for two random ingredients.

	$P(Y = 1)$	$P(Y = 0)$	Total
$P(X = 1)$	$P(i, j)$	$P(i) - P(i, j)$	$P(i)$
$P(X = 0)$	$P(j) - P(i, j)$	$1 - P(i) - P(j) + P(i, j)$	$1 - P(i)$
Total	$P(j)$	$1 - P(j)$	1

From these probabilities, the entropy and the joint entropy can be calculated, which pave the way for calculating mutual information.

The Pearson correlation coefficient, r, named after British scientist Karl Pearson, is a statistical measure used to evaluate the linear relationship between pairs of variables. Its purpose is to help determine whether changes in one variable correspond to changes in another variable. This coefficient ranges between -1 and 1, inclusively, and can be calculated using Eq.3.21. The sign, either positive or negative, indicates the type of relationship, and the absolute value reveals the strength of said relationship. A value of -1 denotes a perfect negative correlation, 1 signifies a perfect positive correlation, and 0 suggests no linear correlation between the variables [99].

$$r = \frac{\sum_{i=1}^{n}(x_i - \langle x\rangle)(y_i - \langle y\rangle)}{\sqrt{\sum_{i=1}^{n}(x_i - \langle x\rangle)^2 \sum_{i=1}^{n}(y_i - \langle y\rangle)^2}} \tag{3.21}$$

where x_i and y_i are individual data points from the sets X and Y, respectively, $\langle x\rangle$ and $\langle y\rangle$ stand for the means of the sets X and Y and n signifies the number of data points.

Since the arithmetic mean is given as $\langle x\rangle = \frac{1}{n}\sum_{i=1}^{n} x_i$ [100], the expression in Eq.3.21 can be simplified , as shown in Eq.3.22.

$$r = \frac{\langle xy\rangle - \langle x\rangle\langle y\rangle}{\sqrt{(\langle x^2\rangle - \langle x\rangle^2)(\langle y^2\rangle - \langle y\rangle^2)}} \tag{3.22}$$

Furthermore, in the case of recipes, since x and y are binary variables, $\langle x^2\rangle = \langle x\rangle$ and $\langle y^2\rangle = \langle y\rangle$. Also, $\langle xy\rangle$ can be calculated according to Eq.3.20, and $\langle x\rangle$ and $\langle y\rangle$ can be calculated according to Eq.3.20. Therefore, in this case, the Pearson correlation coefficient is calculated based on those probabilities, as represented by Eq.3.23.

$$r = \frac{P(i, j) - P(i)P(j)}{\sqrt{(P(i) - P(i)^2)(P(j) - P(j)^2)}} \tag{3.23}$$

3.5 Hierarchical Clustering

Hierarchical clustering is a widely used technique in data exploratory analysis and machine learning that groups similar data points into clusters based on their similarities. The primary idea behind hierarchical clustering is to create a tree-like structure, known as a dendrogram, that illustrates the relationships between data points or clusters [101].

The process involves a sequence of steps, beginning with the calculation of the distance matrix, which represents the dissimilarity between two data points. In this work, the objective is to perform hierarchical clustering based on the correlation matrix **C**. However, this is not a valid distance metric, since it demonstrates similarity, instead of dissimilarity, meaning that the correlation coefficient needs to be converted into a distance measure, which can be done using the formula in Eq.3.24.

$$\mathbf{D} = 1 - \mathbf{C} \tag{3.24}$$

Here, **D** represents the distance matrix, where higher values signify greater dissimilarity between data points [102].

The subsequent action entails selecting a linkage criterion, an important decision that greatly influences the results. There exists a wide array of criteria, each defining the distance between clusters in different ways. While there are numerous options available, this discussion will focus on four of the most common. The first is the *single* linkage, where the distance between two clusters is the shortest distance between any two points in the two clusters. Another common criterion is the *complete* linkage, where the distance is defined as the longest distance between any two points in the clusters. The *average* linkage calculates the distance between clusters as the average distance between all pairs of points in the two clusters. Lastly, there is the *Ward* linkage, chosen in this context, which minimizes the within-cluster variance.

Ward linkage stands out by its ability to minimize the variance within clusters. This is achieved by considering the increase in sum of squared distances when merging clusters. For clusters X_i and X_j, the change in variance upon merging is expressed by Eq.3.25.

$$\Delta(X_i, X_j) = \frac{n_i n_j}{n_i + n_j} \cdot \|c(X_i) - c(X_j)\|^2 \tag{3.25}$$

Where n_i and n_j denote the sizes of clusters X_i and X_j, and $c(X_i)$ and $c(X_j)$ are their respective centroids [101].

Furthermore, the Ward linkage process unfolds through three steps: initialization, agglomeration, and iteration, as seen in Algorithm 4. Initially, each data point is treated as a separate cluster $X_i = \{x_i\}$, $i = (1, ..., n)$, meaning there are n different clusters. Agglomeration involves merging the pair of clusters that minimize the increase in variance, resulting in $n - 1$ different clusters. Finally, the iteration step involves updating the distances between the newly formed cluster and the remaining clusters and repeating the agglomeration step until there is only one cluster containing all the elements. By pursuing variance reduction, Ward's linkage tends to yield compact and cohesive clusters [103].

Algorithm 4 Ward Linkage Process

```
1: $n \leftarrow$ number of data points
2: for $i = 1$ to $n$ do
3:     $X_i \leftarrow \{x_i\}$                                ▷ Each data point as a separate cluster
4: end for
5: while $|X| > 1$ do
6:     Find clusters $X_i$ and $X_j$ with minimal variance increase      ▷ Based on Eq.3.25
7:     Merge clusters $X_i$ and $X_j$ into a new cluster
8:     Update $X$ by removing $X_i$ and $X_j$ and adding the new cluster
9: end while
```

Chapter 4

Results and Analysis

In this chapter, a characterization of the databases employed will be presented, followed by a thorough analysis of the methods examined in previous chapters. The analysis will focus on evaluating their computational time complexity, assessing their ability to generate recipes that accurately represent the structures of the databases they were trained on, and subjecting them to diverse qualitative tests. The aim is to provide a detailed scientific evaluation of these methods, considering their computational efficiency, capability to capture database structures, and overall performance in different testing scenarios.

For the purpose of analyzing and codifying a recipes' dataset, a binary matrix M with dimensions $n_r \times n_i$ was employed. In this context, n_r is the total count of recipes in the database, with each row of the matrix representing an individual recipe. Moreover, n_i represents the overall number of distinct ingredients within the database, with each column of the matrix corresponding to one ingredient. In a succinct mathematical representation, $M_{r,i}$ assumes a binary state: 1 signifies the presence of ingredient i in recipe r, while 0 denotes the absence of said ingredient within the same recipe.

4.1 Characterization of the Databases

Two distinct databases were employed for model testing. The first database exclusively featured soup recipes, while the second encompassed a broader array of recipes from various regions around the world. This approach allowed for comprehensive evaluation across different recipe types and ensured a robust analysis.

4.1.1 Soup Database

Initially, it was used a database consisting of 688 soup recipes, 410 of which are clear soups and 278 are thick soups, with a total of 564 different ingredients among them. The recipe with fewer ingredients had 3, and the one with the most had 34. However, most of those ingredients were very uncommon, appearing only a few times.

To consider only ingredients with statistical information, any ingredient appearing less than 10 times was removed. By doing this, it was also necessary to eliminate the recipes that lost significance, so only the recipes that still contained at least 3 different ingredients were considered. After this filtering, the database was reduced to 678 recipes with 146 different ingredients. The average number of ingredients in a recipe is 9.84 and the recipe with most ingredients has 28, leaving the database with a distribution of the number of ingredients as shown in Fig.4.1. For this database, the matrix M resulted in a sparse matrix with a fill percentage of 6.74%.

In order to later generate recipes with a realistic number of ingredients, a gamma distribution was fitted, as also shown in Fig.4.1. This distribution starts at the origin ($s = 0$) and is characterized by two parameters: shape (k) and scale (θ) (see Eq.4.1).

$$f(x) = \frac{1}{\theta^k \cdot \Gamma(k)} \cdot (x - s)^{k-1} \cdot e^{-\frac{x-s}{\theta}} \tag{4.1}$$

where x is the random variable, which in this case is the number of ingredients of each recipe.

The parameter k determines the curve's shape, allowing it to capture a wide range of distributions, from exponential to normal-like shapes and θ influences the spread or variability of the distribution [104]. The gamma distribution was chosen due to its flexible shape, however, due to the fact that it starts at the origin, and that was not desirable for this data, the curve is shifted by s, as shown in Eq.4.1. Thus, the distribution parameters are the following: $s = -4.27$, $k = 17.24$ and $\theta = 0.818$.

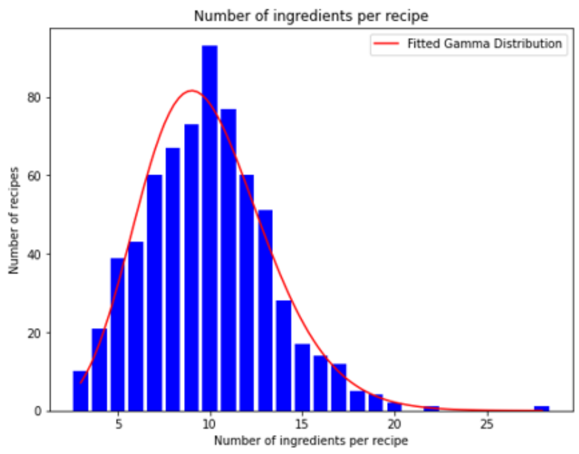


Figure 4.1: Histogram illustrating the distribution of the number of ingredients per recipe in the soup database, along with its corresponding fitted gamma distribution.

Furthermore, in Fig.4.2.a) it is represented the probability of the ingredients being present in a recipe (see Eq.3.19). As it is observable, even after removing the rarest ingredients, the distribution of ingredients among the recipes still leans heavily towards the most common and fundamental ingredients in soups. To improve the clarity in illustrating the probability of the 50 most common ingredients, only those were shown in Fig.4.2.b).

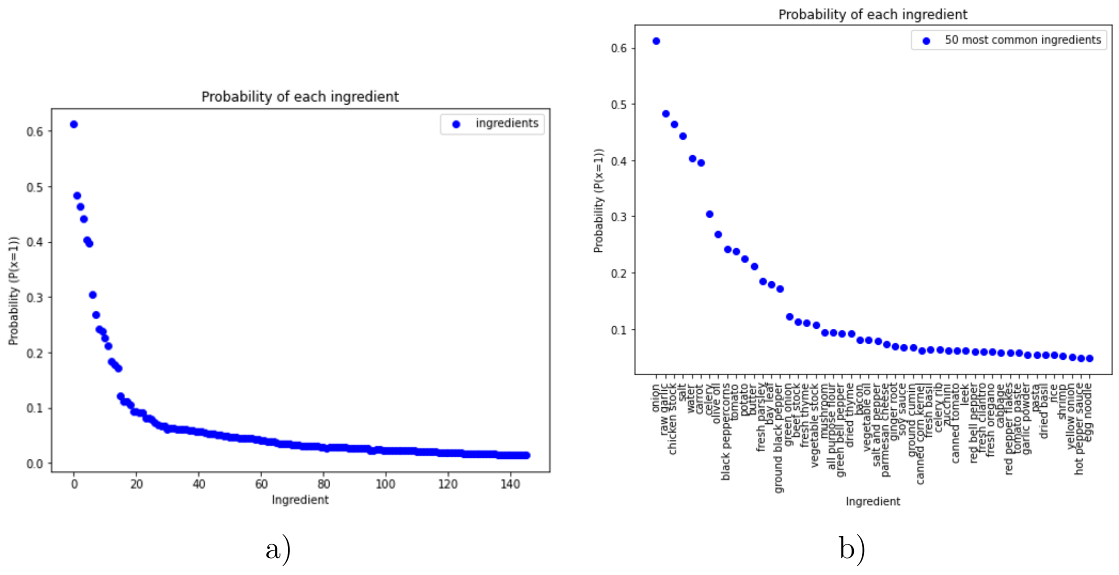


Figure 4.2: Probability of each ingredient being in a recipe, calculated based on the frequency, sorted from most to least common (a), for the soup database, and only for the 50 most common ingredients, featuring their respective names (b).

Afterwords, it was aimed to understand how the mutual information and the correlation coefficient related to one another. In Fig.4.3.a) each point represents one out of the possible combinations, $C_2^{50} = 1225$, formed by the 50 most common ingredients.

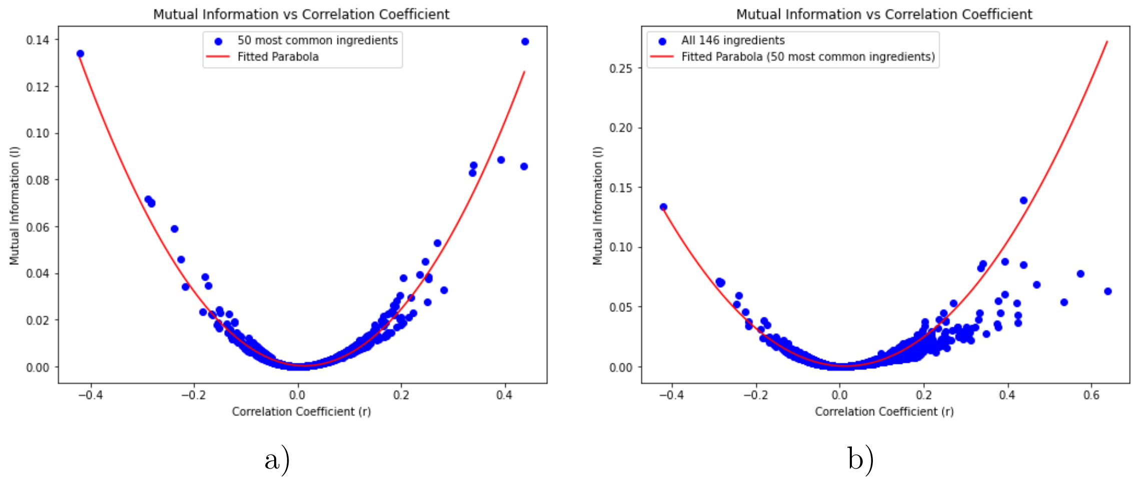


Figure 4.3: Relationship between the mutual information and the correlation coefficient of the pairs of the 50 most common ingredients (a) and the pairs between all of the ingredients (b) for the soup database. It is also represented the fitted parabola of the pairs of the 50 most common ingredients on both images.

It was observed that the relationship between them formed a parabolic shape, where pairs with close to 0 values of r also had close to 0 values of I and pairs with high values of I also had a strong positive or negative correlation. As such, a parabolic fitting was performed, showing that $I = 6.98 \times 10^{-1} r^2 - 1.86 \times 10^{-2} r + 3.79 \times 10^{-4}$. However, analyzing Fig.4.3.b), where all the pairs are represented, and applying the exact same fitting, following the equation mentioned, it is observed that some pairs with less common ingredients had a relatively strong correlation, but due to their lack of appearance, their mutual information was low when compared with pairs of only common ingredients with the same correlation. This means that for less frequent ingredients the mutual information can be misleading.

4.1.2 World Cuisine Database

Next, it was considered the extensive dataset of [14] consisting of 56498 recipes of 11 different cuisines: 352 African, 2512 East Asian, 381 Eastern European, 2917 Latin American, 645 Middle Eastern, 41524 North American, 250 Northern European, 621 South Asian, 457 Southeast Asian, 4180 Southern European and 2659 Western European. This database also had 381 distinct ingredients but, similarly to the other database, the less common ingredients, in this case the ones appearing in less than 200 recipes, were excluded and, following that, the recipes with less than 3 ingredients were also eliminated, leaving a database of 54604 recipes with 190 different ingredients. The average number of ingredients per recipe is 8.25 and the recipe with most ingredients has 30. For this database, M resulted in a sparse matrix with a fill percentage of 4.34%.

As detailed in the preceding section, a gamma distribution was fitted to the distribution of the number of ingredients, as shown in Fig.4.4. The distribution parameters are the following: $s = 1.62$, $k = 3.66$ and $\theta = 1.81$.

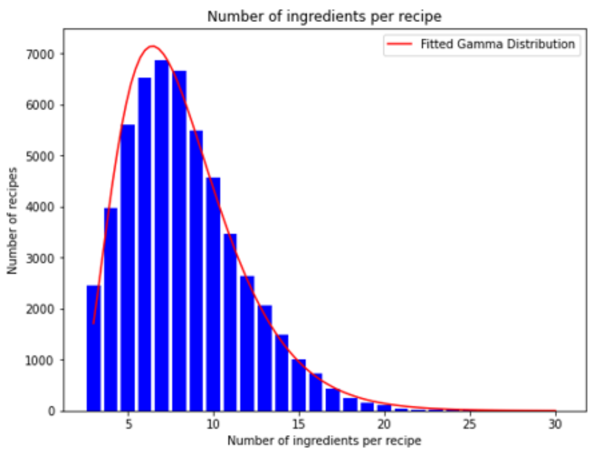


Figure 4.4: Histogram illustrating the distribution of the number of ingredients per recipe in the world cuisine database, along with its corresponding fitted gamma distribution.

The probability of each ingredient being present in a recipe is shown in Fig.4.5. Similarly to the soup database, the distribution of ingredients leans heavily towards the most common ingredients.

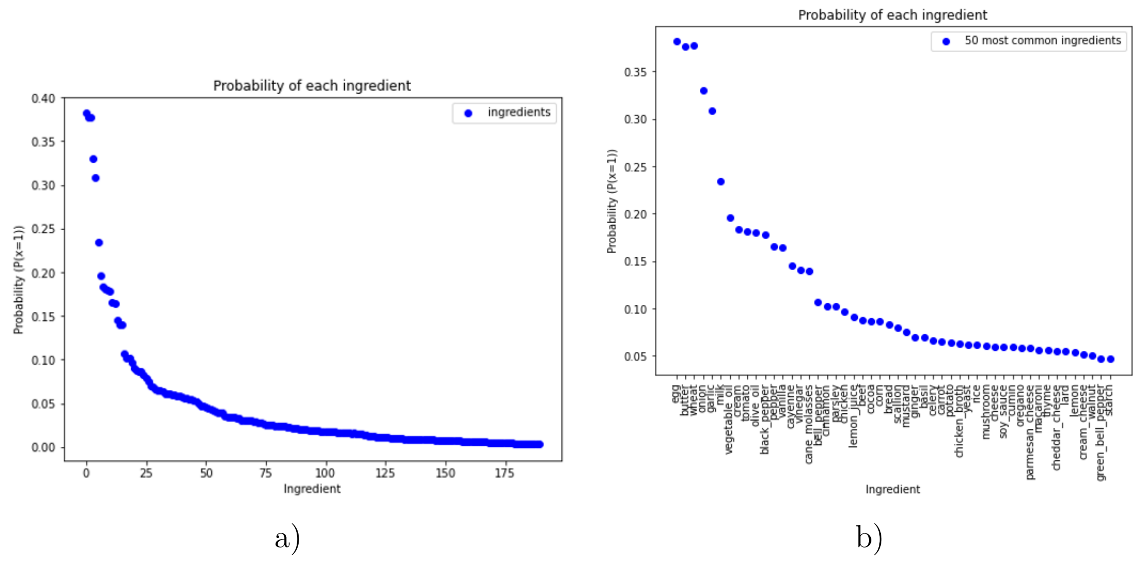


Figure 4.5: Probability of each ingredient being in a recipe, calculated based on the frequency, sorted from most to least common (a), for the world cuisine database, and only for the 50 most common ingredients, featuring their respective names (b).

Finally, regarding the mutual information and Pearson correlation coefficient, this database had a very similar behavior to the other, where a parabolic shape can be seen for the 50 most common ingredients (Fig.4.6). In this case $I = 7.25 \times 10^{-1} r^2 - 2.43 \times 10^{-2} r + 1.39 \times 10^{-4}$.

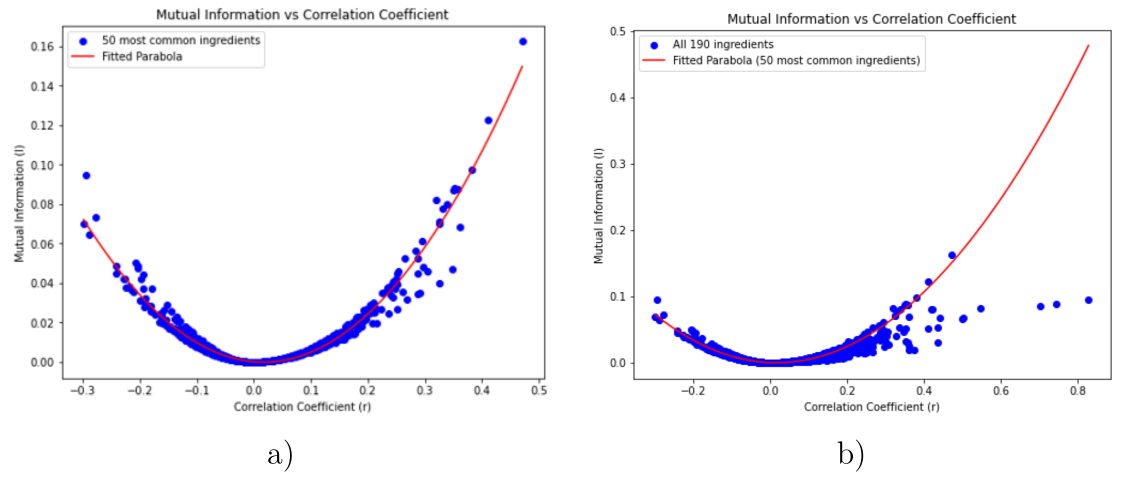
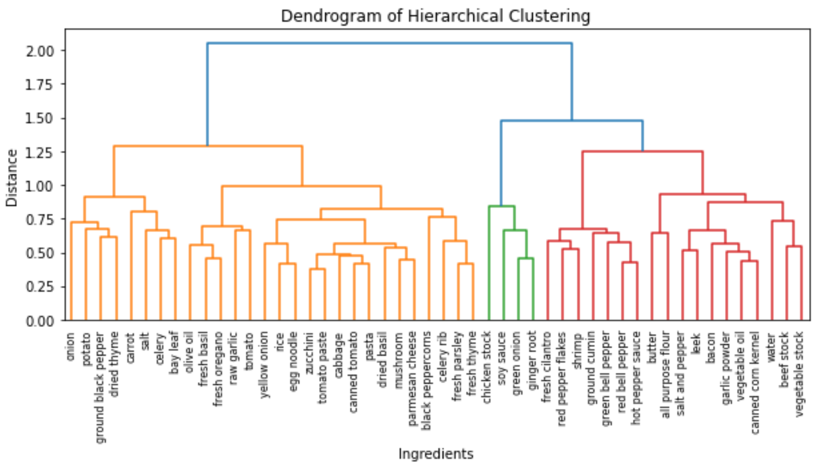


Figure 4.6: Relationship between the mutual information and the correlation coefficient of the pairs of the 50 most common ingredients (a) and the pairs between all of the ingredients (b) for the world cuisine database. It is also represented the fitted parabola of the pairs of the 50 most common ingredients on both images.

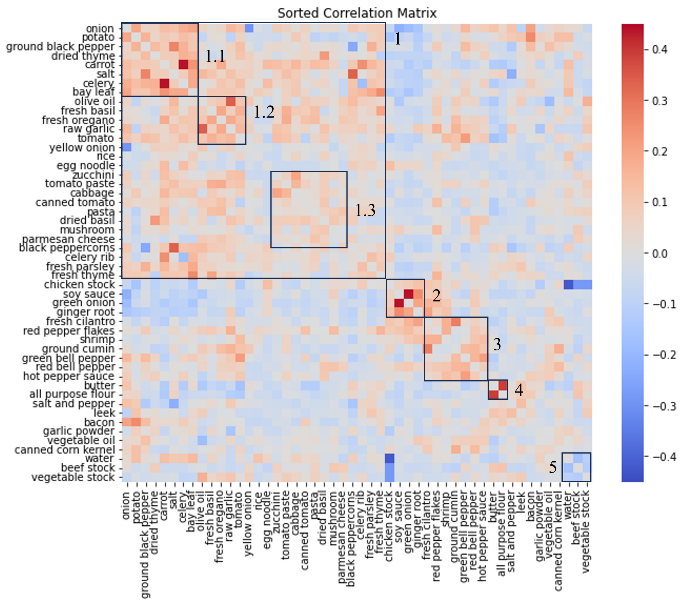
4.1.3 Hierarchical Clustering

After calculating the correlation matrix, it was performed hierarchical clustering for the 50 most common ingredients based on the correlation matrix **C**, in order to further understand the similarity relationships between groups of ingredients.

The information presented in Fig.4.7 (referring to the soup database) and Fig.4.8 (referring to the world cuisine database) showcase the outcomes of three analyses: hierarchical clustering in (a), the respectively ordered correlation matrix in (b) and, from the dendogram and the correlation matrix, the identification of well-defined clusters for both databases, as shown in (c).



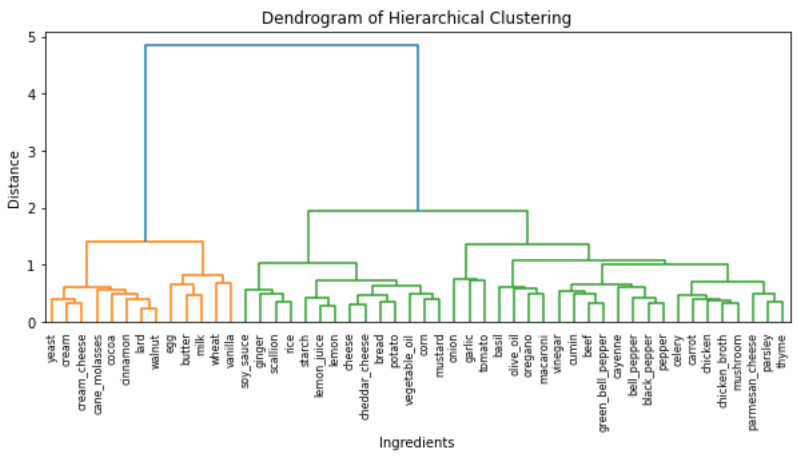
a)



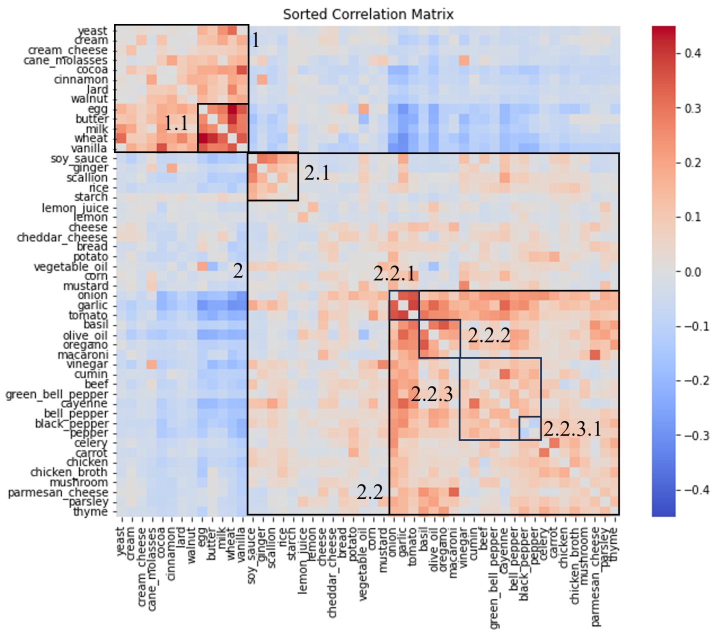
Cluster Number	Cluster Name
1	South European
1.1	Basics
1.2	Mediterranean base
1.3	Italian
2	Asian
3	Latin American
4	Beurre Manié
5	Liquids (negative)

b) c)

Figure 4.7: Dendrogram representing hierarchical clustering (a), ordered Pearson correlation matrix with visual depiction of defined clusters (b), and the corresponding nomenclature of these clusters (c) in the soup database.



a)



b)

Cluster Number	Cluster Name
1	Sweets
1.1	Base of sweets
2	Savory
2.1	Asian
2.2	South Western cuisine
2.2.1	Basics
2.2.2	Italian
2.2.3	Latin American
2.2.3.1	Pepper (negative)

c)

Figure 4.8: Dendrogram representing hierarchical clustering (a), ordered Pearson correlation matrix with visual depiction of defined clusters (b), and the corresponding nomenclature of these clusters (c) in the world cuisine database.

This hierarchical clustering analysis employs the correlation coefficient to examine the relationships among the top 50 most frequently occurring ingredients within the two original databases.

It is also important to mention that the Pearson correlation coefficient of an ingredient with itself is always 1, as per Eq.3.22. Typically, correlation matrices are displayed within a range of -1 to 1, as demonstrated in [105]. However, the objective of this study is to identify clusters and in both databases the highest absolute value was around 0.45. To enhance the visualization and focus on the correlations pertinent to this research, the limits of the correlation matrix were adjusted from -0.45 to 0.45

instead of the conventional -1 to 1 range.

In light of this, and recognizing that self-correlations hold no significance in cluster analysis, all self-correlation values were set to 0, rendering them as gray squares in the visualization. Consequently, blue squares now exclusively represent negative correlations, red squares represent positive correlations, and gray squares denote a lack of correlation between ingredients.

4.2 Computational Time Complexity

In this subsection, the time complexity of the methods employed is studied and analyzed, as seen in Tab.4.1. For each method, this process was divided into two phases. First, there is the training phase, involving the derivation of the transition matrix, matrix X (see Eq.3.14), and the weight and bias matrices for the Markov, NMF, and RBM methods, respectively. This training step occurs only once. Subsequently, there is the suggestion phase, which utilizes the aforementioned matrices to generate suggestions. This operation is performed every time a new completion is generated.

For the non-Markovian chain, the training process starts with initializing the transition matrix with 0 in every entry, requiring $O(n_i^2)$ time. Following this, all possible combinations of two ingredients are generated, iterated through, and checked against each recipe, resulting in a time complexity of $O(C_2^{n_i} \times n_r) = O(n_i^2 \times n_r)$. Normalizing the matrix follows, taking $O(n_i^2)$ time. Thus, the overall time complexity for this process is $O(n_i^2 + n_i^2 \times n_r + n_i^2)$, which can be simplified to $O(n_i^2 \times n_r)$. In the suggestion phase, the columns of the transition matrix corresponding to the ingredients in the current recipe are added and normalized, taking $O(n_p \times n_i)$ time, where n_p is the number of ingredients that the recipe currently has. In the worst case, this process requires $O(n_i^2)$ time. This is repeated until p ingredients are added, making the total complexity $O(n_i^2 \times p \times n_s)$ for generating n_s samples.

In the case of the RBM algorithm, the main computation of the training process occurs within the epochs loop, requiring $O(n_e)$ time, where n_e represents the number of epochs. Within this loop, data is iterated over n_r/b batches (b being the batch size), taking $O(n_r/b)$ time. Furthermore, updating the weight matrix, visible bias, and hidden bias respectively demand $O(n_i \times n_h)$, $O(n_i)$, and $O(n_h)$ time, with n_i, the number ingredients, being the number of visible units and n_h being the number of hidden units. Consequently, the total complexity becomes $O(n_e \times n_r/b \times (n_i \times n_h + n_i + n_h))$, which simplifies to $O(n_e \times n_r/b \times n_i \times n_h)$. In the suggestion process of this model, the state of the hidden layer is calculated first. This is achieved by multiplying the current visible layer with the weight matrix and adding the hidden bias. Subsequently, the activation function (sigmoid) is applied. The time complexity for this operation in recipe is $O(n_i \times n_h + n_h)$, which can be simplified to $O(n_i \times n_h)$. Next, to calculate the probabilities of activation of the visible units based on these state of the hidden layer, the inverse process is employed: multiplying the hidden layer with the transpose of the weight matrix, adding the visible bias, and applying the sigmoid function again. This step also has a time complexity of $O(n_i \times n_h)$ for each recipe. Considering these complexities, suggesting a single ingredient takes $O(n_i \times n_h)$ time. If p ingredients need to be suggested for the recipe, the time complexity becomes $O(p \times n_i \times n_h)$.

Regarding the NMF algorithm, the training process first involves calculating matrices W and H. The time complexity of one iteration of this operation is $O(n_i n_r k)$, where k represents the number of features. Overall, the time complexity, in the worst case, can be expressed as $O(n_i n_r k \times \max_{\text{iter}})$, with $\max_{\text{iter}}$ indicating the maximum number of iterations until convergence. Additionally, matrix X needs to be calculated. This process includes inverting matrix M $(O(n_r^2 n_i))$, multiplying it with W $(O(n_r^2 k))$, and then further multiplying the result with H $(O(n_r k n_i))$. Consequently, the training process's time complexity is $O(n_i n_r k \times max_{iter} + n_r^2 n_i + n_r^2 k + n_r k n_i)$, which can be simplified to $O(n_i n_r k \times max_{iter} + n_r^2 n_i + n_r^2 k)$. Moving on to the suggestion phase, where incomplete recipes r are multiplied with matrix X, each recipe r incurs a time complexity of $O(n_i^2)$. This allows the suggestion of a single ingredient. To suggest p ingredients for the recipe, the time complexity becomes $O(p \times n_i^2)$.

Table 4.1: Time complexity comparison of the different methods during the training phase and for single-ingredient suggestions.

	Markov	RBM	NMF
Training	$O(n_i^2 \times n_r)$	$O(n_e \times n_r/b \times n_i \times n_h)$	$O(n_i n_r k \times max_{iter} + n_r^2 n_i + n_r^2 k)$
Suggesting	$O(n_i^2 \times n_s)$	$O(n_i \times n_h)$	$O(n_i^2)$

Regarding the training process, the Markov and RBM models demonstrate superior efficiency in training compared to the NMF algorithm, especially for extensive recipe databases. This disparity arises from the NMF algorithm's time complexity, which is proportional to n_r^2, while for Markov and RBM models, it depends solely on n_r. Consequently, in scenarios involving large databases, NMF might not be the optimal choice due to its higher computational demands.

When it comes to suggesting, RBM stands out as the most efficient method. This efficiency is due to the relatively smaller number of hidden units (n_h) compared to visible units (n_i), making $n_i \times n_h$ smaller than n_i^2. Conversely, the Markov-based model lags behind, requiring a substantial number of samples to establish a significant statistical ensemble. This inefficiency can be a critical concern, especially in real-time applications where quick responses are essential. Consequently, the Markov-based model might not be the best choice for time-sensitive applications.

4.3 Recipe Generation

In order to generate recipes using the methods described, there is the need to first select a starting ingredient. In this work, to determine such ingredient for each simulation, the frequency distribution of the ingredients was consulted. Selecting the initial ingredient based on frequencies ensures a higher probability of generating coherent and realistic recipes, as it reflects the prevalence of ingredients in the database. Then, the probability of selecting the initial ingredient was computed using Eq.4.2.

$$P(\text{initial ing}) = \frac{\sum_{r=1}^{n_r} S_r(\text{initial ing})}{\sum_{j=1}^{n_i} \sum_{r=1}^{n_r} S_r(j)} \tag{4.2}$$

In this context, n_r denotes the total number of recipes, n_i stands for the count of distinct ingredients, and S_r signifies the presence or absence of an ingredient in recipe r.

To ensure a realistic range of ingredients within the simulated recipes, the gamma distribution depicted previously (in Fig.4.1 and Fig.4.4) was employed. For each simulated recipe, a random number was generated using said gamma distribution. Nevertheless, only values between 3 and 25 were considered to maintain a plausible ingredient count. Any numbers outside this range were disregarded, and a new random number was generated instead.

In the context of recipe generation, the organized list of suggestions associated with each method (Y_r) was taken into consideration. This list of suggestions was normalized and, subsequently, a random ingredient was selected to be included in the recipe based on probabilities associated with normalized Y_r. This sequence of steps was iteratively applied to each subsequent ingredient incorporated into the recipe, continuing until the recipe attained the predetermined number of ingredients previously defined.

Then, to evaluate the accuracy of the employed methods in capturing relationships and correlations among different ingredients, a series of tests was conducted.

Initially, 1000 recipes were simulated using the NMF and RBM method trained for the soup database, in order to choose the most appropriate parameters. This database was chosen since its smaller size allowed for multiple repetitions in a reasonable time. Then, for the world cuisine database, the same parameters were implemented. In order to achieve this, the mutual information and the correlation coefficient were taken into account. The database generated by each method had an average of 9.993 ingredients per recipe, the shortest recipe had 3 ingredients and the longest one had 23.

To differentiate between the mutual information of pairs with a robust interdependency, where the presence of one ingredient implies the presence of the other, and pairs where the presence of one ingredient suggests the absence of the other, an adjustment was made to the mutual information of pairs exhibiting a negative correlation coefficient. This adjustment involved multiplying the mutual information of negatively correlated pairs by -1, resulting in a weighted mutual information denoted as I'.

Finally, the Euclidean distance between the weighted mutual information and correlation coefficient values of pairs in the initial database and those in the simulated recipe database was calculated. The Euclidean distance, denoted as d_E, quantifies the similarity or dissimilarity between data points in a multi-dimensional space [106]. The Euclidean distance is determined using Eq.4.3.

$$d_E = \sqrt{\sum_{i=1}^{ni} \sum_{j=i+1}^{ni} (I'^d_{ij} - I'^s_{ij})^2} \tag{4.3}$$

where I'^d_{ij} represents the weighted mutual information between the pair of ingredients i and j in the original database and I'^s_{ij} representing the weighted mutual information between the pair of ingredients i and j in the simulated recipe database.

In this case, the space possesses $C_2^{ni} = \frac{ni!}{2!(ni-2)!}$ dimensions, where each dimension corresponds to one of the combinations of two between the different ingredients, and is represented by a pair of coordinates (I'^d_{ij}, I'^s_{ij}). The same principle can be applied for the correlation coefficient.

This methodology permits the quantification of dissimilarity between the real databases and the simulated ones. Thus, the parameters minimizing the Euclidean distance of the weighted mutual information and the correlation coefficient were selected.

For the NMF algorithm it was the parameter $k = 2$ and for the RBM, the parameters that better optimized the generated recipes were: 40 hidden units, learning rate $\alpha = 0.1$ and 100 epochs with a batch size of 10.

After training the models, in order to have a big enough statistical ensemble, instead of generating only 1000 recipes, 10000 recipes were generated for each of the models based on both databases.

To ensure a fair comparison of results across different methods, consistent initial ingredients and the same number of ingredients per recipe were maintained in the datasets generated by each method. When training the methods with the soup database, the average number of ingredients per recipe was 9.85. The recipes ranged from a minimum of 3 ingredients to a maximum of 25. Similarly, in the case of methods trained on the world cuisine database, the average number of ingredients per recipe was 8.26, with recipes containing as few as 3 ingredients and as many as 25.

4.3.1 Recipe Diversity

Initially, the primary objective was to assess the uniqueness of the generated recipes and identify any potential repetitions. Subsequently, the analysis aimed to discern whether these unique recipes merely replicated existing ones in the database or if they constituted entirely new combinations. The results of this analysis can be found in Tab. 4.2.

Table 4.2: Percentage of unique recipes among the total generated (in the first row), and the percentage of these unique recipes that replicate existing database entries (in the last row).

	Soup Database			World Cuisine Database		
	Markov	RBM	NMF	Markov	RBM	NMF
Unique recipes	29.33%	99.98%	99.99%	36.21%	99.02%	99.80%
Database entries	0.07%	0%	0.01%	5.27%	3.71%	1.06%

In Tab.4.2, it is evident that the RBM and NMF generated a wide variety of recipes, incorporating diverse combinations of ingredients. In contrast, the non-Markovian chain algorithm exhibits a notable pattern of repetition, utilizing the same ingredient combinations repeatedly. This pattern aligns with the findings depicted in Fig.4.9.a) and Fig.4.10.a), presented in the following section. The Markov algorithm tends to rely heavily on suggesting common ingredients, resulting in duplicate recipes without the creativity and variety found in RBM and NMF outputs.

When it comes to replicating existing recipes, all the methods consistently displayed a higher propensity for reproducing recipes from the world cuisine database. This particular database, in contrast to the soup database, contains a significantly larger number of recipes, even though the number of ingredients remains roughly similar. As a result, the potential for generating unique recipes is notably reduced when training the methods with the larger database. Furthermore, the Markov algorithm displayed a more pronounced tendency to duplicate recipes from the databases, further corroborating that the Markov algorithm leans towards repeating specific combinations of ingredients.

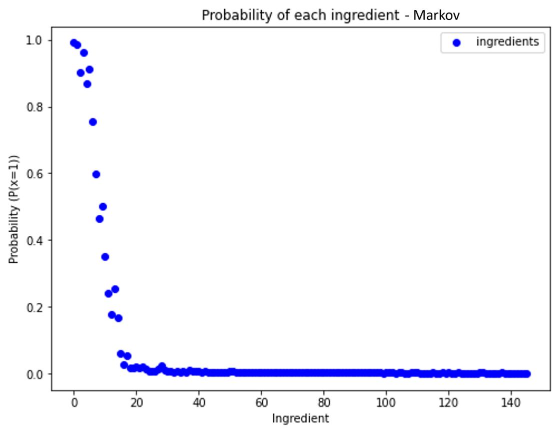
4.3.2 1-Dimensional Analysis: Frequency

For control purposes, a random suggestion based on the distribution of the ingredients (Fig.4.2 and Fig.4.5), which is going to be referred to as the RNG (Random Number Generator) or random method, was also implemented. For this method, Y_r was considered to be the number of recipes in which each ingredient is present.

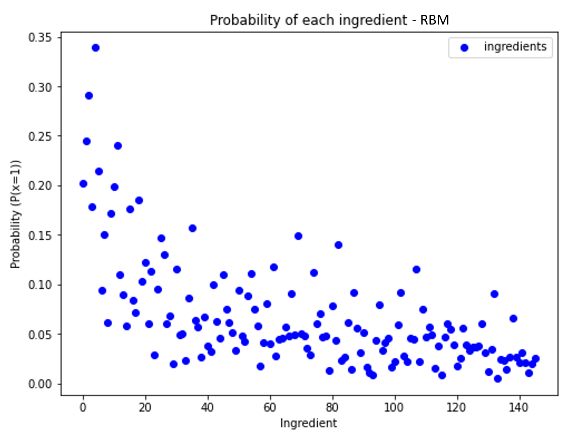
A 1-Dimensional analysis, referencing the probability of an ingredient being present in a recipe (see Eq.3.19), was first conducted. For this, the plots of said probabilities for each ingredient were made and the Euclidean distance between the probability of the ingredients in the original databases and in the simulated databases was calculated according to Eq.4.4. The graphical representations of these analyses are presented in Fig.4.9 and Fig.4.10, providing visual insights into the probability distributions across the datasets generated by each method.

$$d_E = \sqrt{\sum_{i=1}^{ni} (P^d(i) - P^s(i))^2} \tag{4.4}$$

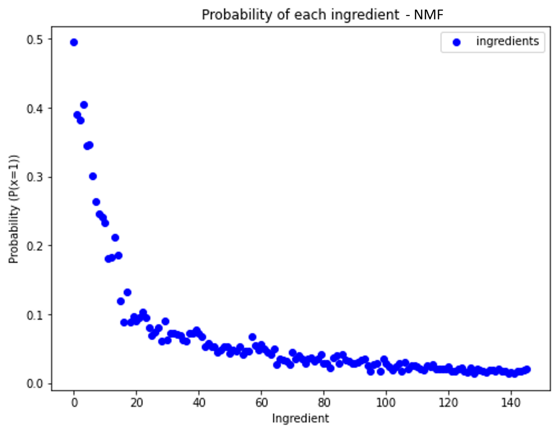
where $P^d(i)$ represents the probability of ingredient i being present in a recipe of the original dataset and $P^s(i)$ represents the probability of ingredient i being present in a recipe of the simulated dataset.



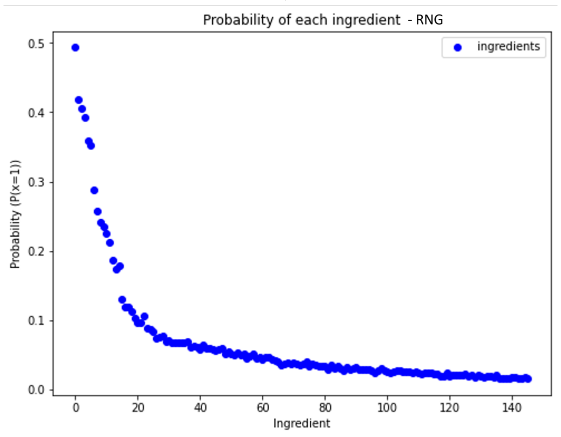
a)



b)

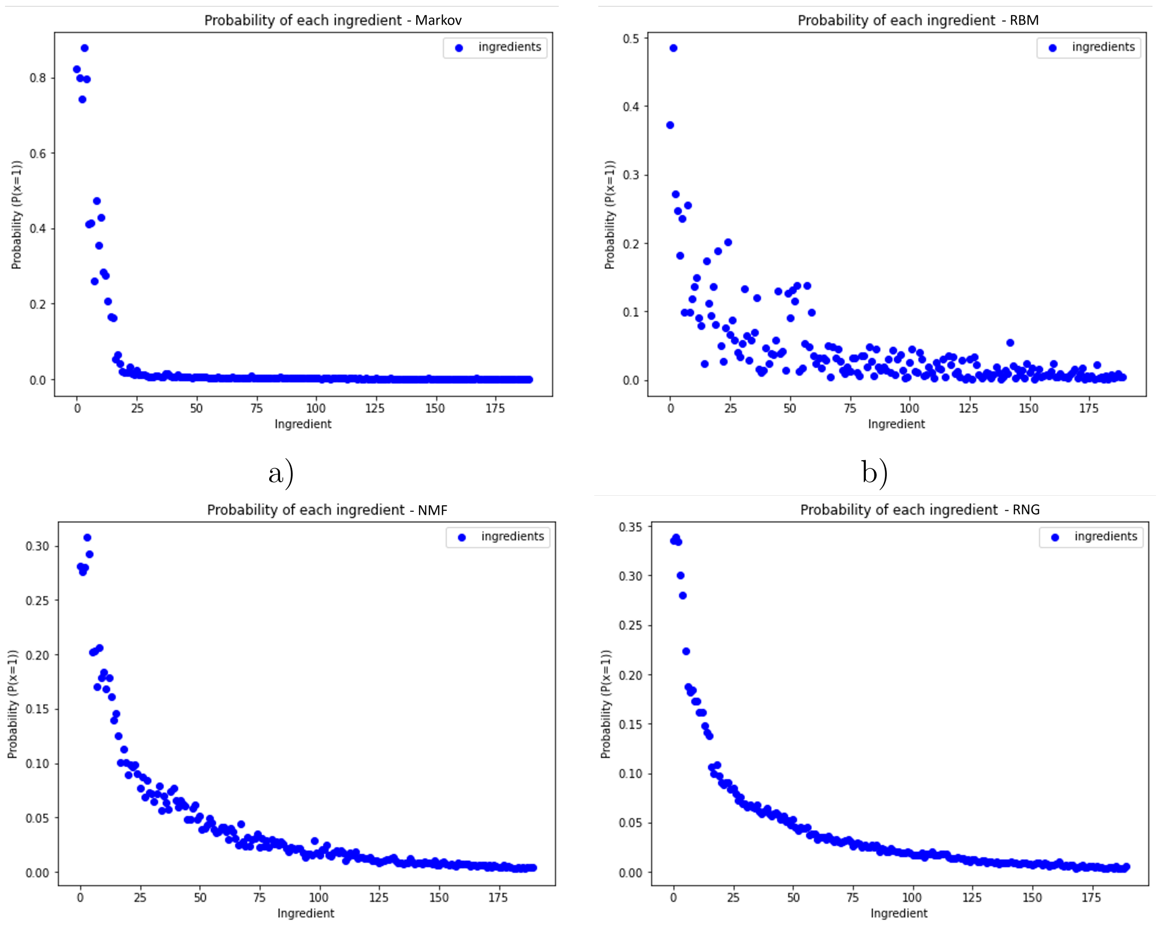


c)



d)

Figure 4.9: Representation of the probability of each ingredient being in a recipe, calculated based on the frequency, for the databases generated by the Markov (a), RBM (b), NMF (c) and RNG (d) algorithms trained on the soup database. The ingredients are sorted from the most to the least common in the original database.



a) b) c) d)

Figure 4.10: Representation of the probability of each ingredient being in a recipe, calculated based on the frequency, for the databases generated by the Markov (a), RBM (b), NMF (c) and RNG (d) algorithms trained on the world cuisine database. The ingredients are sorted from the most to the least common in the original database.

Comparing Fig.4.9 with Fig.4.2.a) and Fig.4.10 with Fig.4.5.a), it is evident that NMF closely matches the real ingredient distribution, maintaining the probability of finding specific ingredients in recipes. On the other hand, the Markov method favors common ingredients, which are present in almost every recipe. This explains the high number of repeated recipes seen previously.

Regarding the RBM, the probability distribution does not follow an order similar to the real one. In this method, some very common ingredients in the databases are rarely observed, which is specially visible for the soup database. In the results corresponding to this database, even some not so common ingredients can be seen multiple times in the recipes. This indicates that RBM does not prioritize ingredient probabilities, suggesting a preference for capturing other underlying patterns rather than mirroring the actual ingredient frequencies. Finally, as expected, the RNG algorithm closely mirrors the original distribution, given that it is exclusively derived from it.

The results from the Euclidean distance analysis further validate the observations mentioned above, as demonstrated in Tab.4.3.

Table 4.3: Euclidean distance between the probability of an ingredient being in a recipe of the original database $P^d(i)$ and the simulated database $P^s(i)$ for each method for both databases.

	Soup Database				World Cuisine Database			
	Markov	RBM	NMF	RNG	Markov	RBM	NMF	RNG
$d_E(P^d(i), P^s(i))$	1.40	0.79	0.22	0.18	1.22	0.46	0.19	0.09

4.3.3 2-Dimensional Analysis: Correlations

In this part of the work, a 2-Dimensional analysis is done, where the correlation coefficient and weighted mutual information of pairs of ingredients are calculated and compared. The correlation coefficient is also calculated separately for the ingredient pairs with positive and negative correlations, in order to understand in which case the methods perform better.

In Tab.4.4 and Tab.4.5 the results of the Euclidean distances calculated for the different methods used in this study are displayed. These results reveal that the most accurate outcomes were achieved with the RBM method, while the modified Markov algorithm had the least accurate results.

Table 4.4: Euclidean distance between the weighted mutual information (I') and the correlation coefficient r of every pair of ingredients, as well as the correlation coefficient for the pairs of ingredients with positive ($r > 0$) and negative ($r < 0$) correlations, for each method for the soup database.

	$d_E(I'^d, I'^s)$	$d_E(r^d, r^s)$	$d_E(r^d, r^s), r > 0$	$d_E(r^d, r^s), r < 0$
Markov	0.91	5.76	4.88	3.05
RBM	0.38	4.06	3.19	2.51
NMF	0.52	5.83	4.78	3.33
Random	0.54	6.01	4.97	3.39

Table 4.5: Euclidean distance between the weighted mutual information (I') and the correlation coefficient r of every pair of ingredients, as well as the correlation coefficient for the pairs of ingredients with positive ($r > 0$) and negative ($r < 0$) correlations, for each method for the world cuisine database.

	$d_E(I'^d, I'^s)$	$d_E(r^d, r^s)$	$d_E(r^d, r^s), r > 0$	$d_E(r^d, r^s), r < 0$
Markov	1.04	4.74	4.14	2.31
RBM	0.51	4.25	3.79	2.04
NMF	0.49	4.71	4.14	2.25
Random	0.64	5.90	4.89	3.31

Importantly, all methods outperformed the RNG in terms of the correlation coefficient. This indicates that these methods effectively capture the correlations between ingredients, surpassing random chance.

However, the Markov algorithm had a higher Euclidean distance than the RNG for the weighted mutual information for both databases. This discrepancy can be attributed to the low values of mutual information in the databases. The Markov algorithm tends to repeat the same ingredient combinations and recipes multiple times, leading to artificially inflated mutual information values. The correlation coefficients, although inflated, are closer to the actual values, since these have a higher absolute value, compared to the mutual information results.

The heat map of the correlation matrix for the 50 most common ingredients, as seen in Fig.4.7.b) and Fig.4.8.b) for the databases, was also made for these methods keeping the order obtained in the hierarchical clustering of Fig.4.7.a) and Fig.4.8.a) and the same limits of -0.45 and 0.45 for the heat scale.

When analyzing the soup database, a comparison between the graphs presented in Fig.4.11 and the reference in Fig.4.7.b) clearly highlights the RBM as the method generating the most similar results. Almost every cluster can be identified to some degree, with both the heat map and the range of correlation values closely resembling the reference, fluctuating between approximately -0.45 and 0.45.

Moreover, the Markov algorithm also exhibited a comparable shape, although clusters 1.1 and 1.2 appeared somewhat merged. Notably, certain positive correlations were excessively strong, surpassing a maximum value of 0.6, while crucial negative correlations, such as those between 'chicken stock' and 'water', went undetected.

In contrast, the NMF heat map deviated significantly from the reference. Only cluster 1.1 could be confidently identified, and the correlations among ingredients were weaker than anticipated.

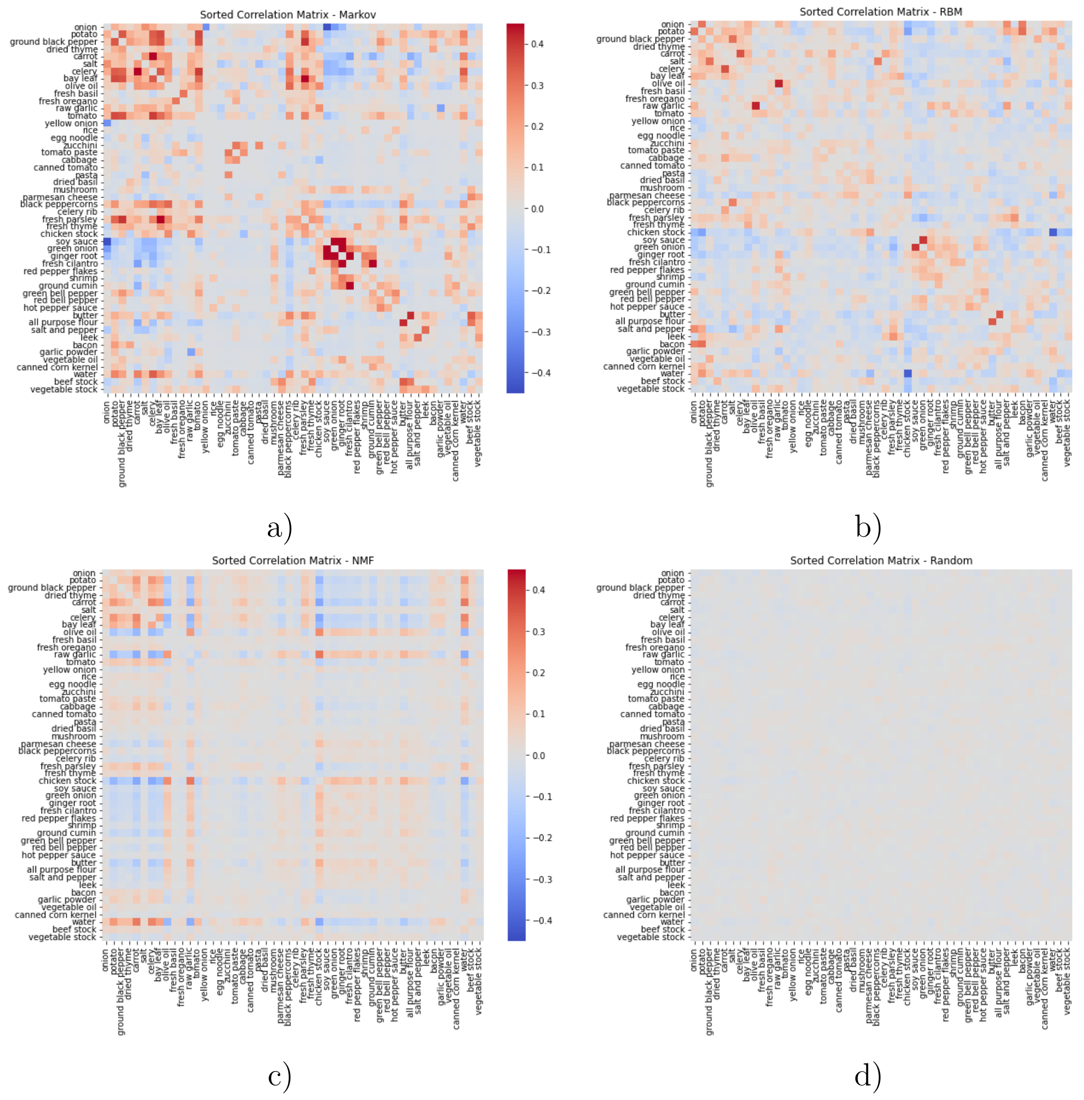


Figure 4.11: Correlation matrix of the 50 most common ingredients for the databases generated by the Markov (a), RBM (b), NMF (c) and RNG (d) algorithms trained on the soup database. The ingredients are sorted according to the hierarchical clustering of the original database.

Examining the world cuisine database, by comparing the graphs in Fig.4.12 with the reference in Fig.4.8.b), a similar result can be observed. The RBM exhibited remarkable similarity with the reference, both in terms of shape and range, indicating a robust alignment between the datasets. However, cluster 2.1 is not identifiable. On the other hand, the Markov algorithm displayed a similar shape, but the correlations, in absolute terms, were considerably stronger than those found within the world cuisine database and the negative cluster 2.2.3.1 can not be identified.

Meanwhile, the NMF algorithm also showed similar clusters with the database, yet it noticeably oversimplified the intricate patterns present. This observation suggests that while the NMF algorithm approximated the basic structure, it failed to capture the nuanced complexities inherent in the culinary dataset. However, in this case the correlations had similar strength to the real one, but some clusters, such as 2.1 and the distinct clusters inside 2.2 can not be detected.

It is also important to mention that the correlations of the recipes generated by the RNG method for both databases, for every pair, were nearly zero. This phenomenon stems from the approach's focus being solely on individual ingredient probabilities, neglecting the complete nature of recipes. This divergence highlights the considered algorithms' ability to capture specific aspects of recipe intricacies, revealing their capacity to discern and reproduce culinary patterns.

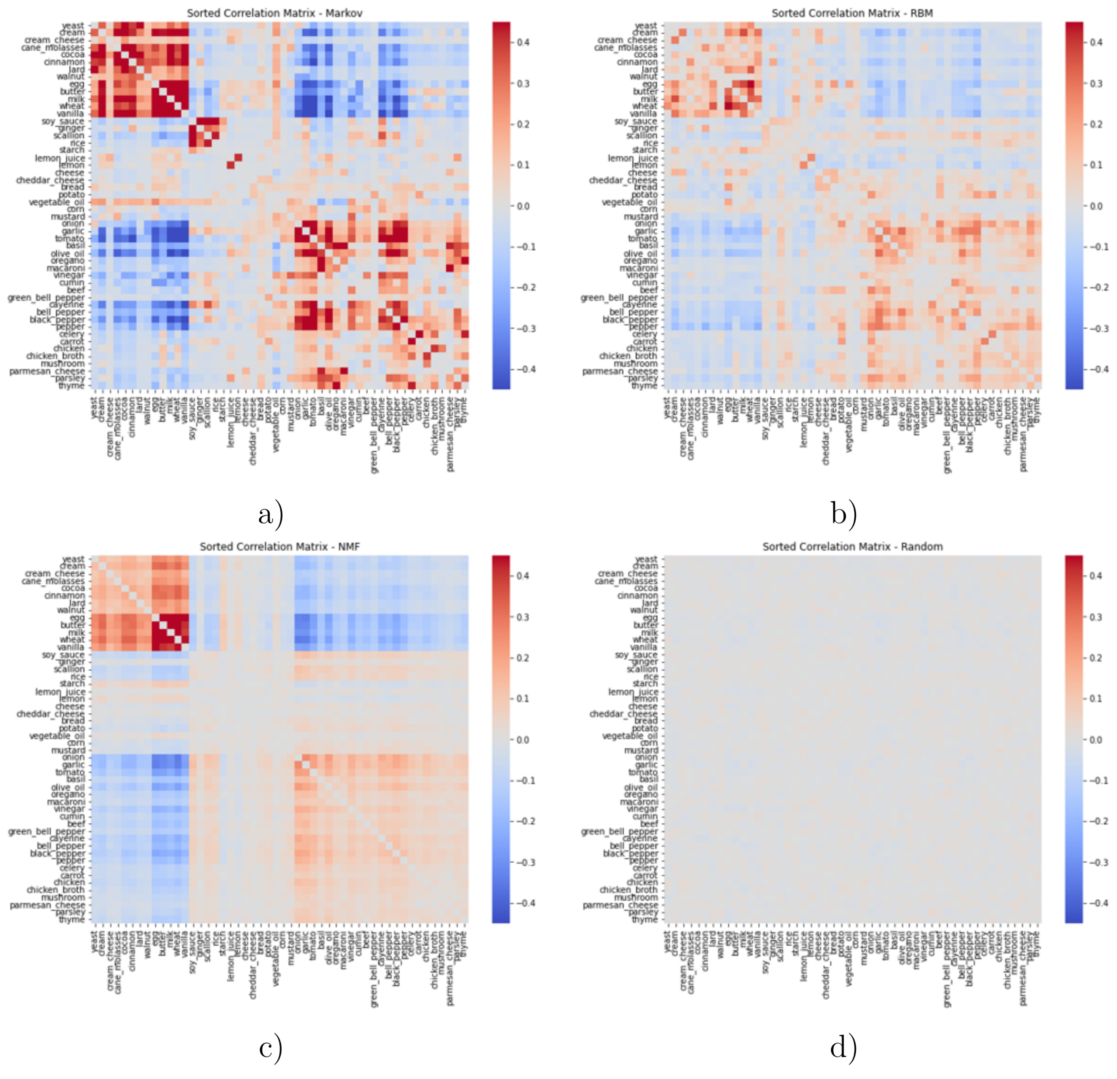


Figure 4.12: Correlation matrix of the 50 most common ingredients for the databases generated by the Markov (a), RBM (b), NMF (c) and RNG (d) algorithms trained on the world cuisine database. The ingredients are sorted according to the hierarchical clustering of the original database.

4.4 Qualitative Analysis

Lastly, a qualitative analysis was conducted, incorporating pertinent examples drawn from the applied methods.

4.4.1 Prohibited Combinations

The analysis begins by examining a few pairs of common ingredients that never appeared together in the databases. The number of times, out of the 10000 recipes generated for each method, that these prohibited ingredient combinations appeared is presented in Tab.4.6 for recipes generated using the soup database to train the models and in Tab.4.7 for recipes generated using the world cuisine database.

Table 4.6: Occurrences of prohibited pairs in 10000 simulated recipes generated by Markov, RBM, and NMF algorithms trained on the soup dataset.

	Onion & Yellow onion	Beef stock & Vegetable stock	Canned corn kernel & Celery rib	Celery & Celery rib
Markov	38	27	0	18
RBM	4	33	14	8
NMF	233	88	49	164

In the analysis presented in Tab.4.6, it is evident that the NMF method exhibited the highest occurrence of prohibited ingredient combinations. For instance, consider a recipe generated by NMF: [onion, water, carrot, celery, potato, beef stock, vegetable stock, dried thyme, cabbage, ground white pepper, frozen split peas, sea salt]. This recipe includes three different types of liquids—water, beef stock, and vegetable stock—which is excessive for typical a soup recipe.

Similarly, an examination of a recipe generated by the Markov method: [onion, raw garlic, chicken stock, salt, water, carrot, celery, olive oil, black peppercorns, tomato, potato, butter, fresh parsley, bay leaf, ground black pepper, fresh thyme, celery rib] reveals certain shortcomings. The Markov method tends to prioritize the most common ingredients found in nearly every recipe (see Fig.4.9.a)). Given that celery is a highly common ingredient, appearing in 7563 recipes, it would be unusual for the Markov method not to pair celery and celery rib together. Moreover, the recipe includes all 15 of the most popular ingredients, from onion to ground black pepper (refer to Fig.4.2.b)), without regard for their compatibility.

Table 4.7: Occurrences of prohibited pairs in 10000 simulated recipes generated by Markov, RBM, and NMF algorithms trained on the world cuisine dataset.

	Vanilla & Olive	Cocoa & Ham	Chicken broth & Strawberry	Gelatin & Pea
Markov	0	0	0	0
RBM	1	0	1	2
NMF	19	27	5	1

In the analysis of the world cuisine database, presented in Tab.4.7, the Markov algorithm did not utilize any of the prohibited ingredient combinations outlined. This phenomenon likely arises because in each pair of ingredients, at least one element is relatively uncommon, since among the 50 most common ingredients there were no prohibited pairs, where two ingredients were never observed together.

Comparatively, when examining the results overall, fewer number of recipes featured both ingredients when compared with the soup database. This disparity in occurrence might be attributed to a few factors. Firstly, the chosen ingredients in this database are less commonplace than those selected for the soup database. Furthermore, the examples provided demonstrate a significant separation between clusters, where one ingredient is conventionally used in sweet dishes and the other in savory ones.

In the soup database, three examples involving two types of onion, celery, and stock were analyzed. These ingredients were not found together because they are interchangeable. This implies that the ingredients that typically pair well with onion also harmonize with yellow onion. In contrast, ingredients like cocoa and ham do not usually complement each other, further facilitating the separation of such ingredients by the algorithms.

Examining specific examples, the recipe where the RBM failed to keep chicken broth and strawberry apart appears to be an attempt to create a savory dish, hindered only by the inclusion of strawberry: [tomato, pepper, bell pepper, mustard, chicken broth, cilantro, orange, seed, dill, strawberry, broccoli]. Conversely, the recipe featuring vanilla and olive represents an incongruous fusion of sweet and savory elements: [vanilla, lemon juice, honey, olive, lettuce, pear]. Similarly, the NMF-generated recipe combining chicken broth and strawberry represents a mismatched combination of sweet and savory elements: [egg, butter, wheat, milk, cream, cane molasses, cinnamon, ginger, carrot, chicken broth, cream cheese, apple, strawberry, feta cheese].

4.4.2 Uncommon Combinations

After conducting this analysis, the focus was on examining the methods' behavior concerning the infrequent interaction between two widely used ingredients. Fig.4.7.b) and Fig.4.8.b) were consulted for this purpose. The analysis specifically focused on a square colored deep blue, indicating a negative correlation between the selected ingredient pairs.

For the soup database, the interaction between water and chicken stock was studied, while in the context of the world cuisine database, attention was given to the combination of onion and vanilla. These specific pairs were chosen due to their rare occurrence in interactions, as illustrated in Tab. 4.8.

Table 4.8: Percentage of recipes, for the three studied methods and the respective training database, that contain chicken stock and water and onion and vanilla, for the soup and world cuisine recipes, respectively.

	Soup database	World cuisine database
	Water & Chicken Stock	Onion & Vanilla
Database	8.30%	0.59%
Markov	81.52%	28.16%
RBM	4.03%	0.04%
NMF	7.65%	1.64%

Examining Tab.4.8, it becomes evident that the Markov algorithm disregards these negative correlations. Specifically, in the case of water and chicken stock, nearly every recipe generated by this algorithm includes both ingredients. Contrarily, the RBM actively avoids pairing these ingredients, demonstrating an even lower frequency of occurrence than in the original databases. This highlights the method's inclination to steer clear of unconventional ingredient combinations.

Additionally, although the NMF method displays these pairs more frequently than the RBM, it remains comparable to the original database, suggesting a deliberate effort to maintain these correlations negative. These findings align with the patterns observed in the heat maps presented in Fig.4.11 and Fig.4.12, reinforcing the consistency of the results across different analytical perspectives.

4.4.3 Recipe Completion

Finally, an incomplete recipe composed of basic ingredients, which required completion, was considered to assess the recommendations generated by each method. The same recipe was utilized to analyze results across both databases, featuring onion, tomato, parsley, beef, and potato. Detailed outcomes can be found in Tab.4.9 and Tab.4.10.

Table 4.9: Top 5 ingredients with the highest probability to be added to an incomplete recipe featuring onion, tomato, parsley, beef, and potato. Each column corresponds to the results obtained from the Markov, RBM and NMF algorithms, respectively, trained using the soup dataset.

Markov		RBM		NMF	
Ingredient	Prob.	Ingredient	Prob.	Ingredient	Prob.
salt	0.061	water	0.153	carrot	0.082
carrot	0.055	salt and pepper	0.086	water	0.079
water	0.051	canned corn kernel	0.041	celery	0.068
raw garlic	0.045	italian seasoning	0.041	salt	0.062
celery	0.044	olive oil	0.039	bay leaf	0.042

Regarding the soup database, it is evident that the considered recipe lacks essential seasonings, particularly salt, and a liquid component. These elements were consistently suggested as the top choices by all three methods. Additionally, the other ingredients listed in the top 5 probabilities appear to be sensible additions to the recipe.

Notably, in the RBM method, the probabilities of suggesting the first two ingredients are significantly higher compared to the probabilities of suggesting any other ingredient. In contrast, the Markov method, and to some extent the NMF, exhibit probabilities that are more evenly distributed among the top suggestions. This discrepancy suggests that the RBM method provides suggestions with a higher level of certainty, emphasizing specific ingredients with greater confidence.

Table 4.10: Top 5 ingredients with the highest probability to be added to an incomplete recipe featuring onion, tomato, parsley, beef, and potato. Each column corresponds to the results obtained from the Markov, RBM and NMF algorithms, respectively, trained using the world cuisine dataset.

Markov		RBM		NMF	
Ingredient	Prob.	Ingredient	Prob.	Ingredient	Prob.
garlic	0.077	black pepper	0.134	garlic	0.086
pepper	0.045	carrot	0.101	olive oil	0.046
black pepper	0.040	garlic	0.080	black pepper	0.044
butter	0.035	bacon	0.058	cayenne	0.039
egg	0.033	pepper	0.041	pepper	0.038

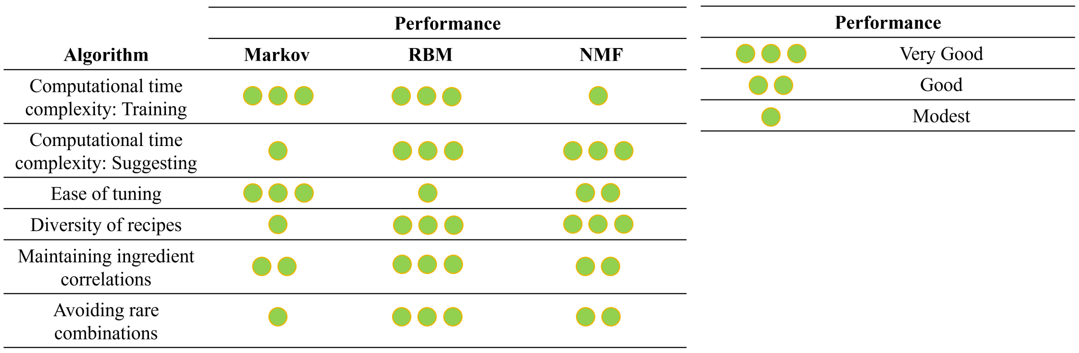
In the context of the world cuisine database, determining the most suitable additions to the recipe is a complex task due to the diverse range of cuisines and dishes considered. However, one consistent observation is the need for seasoning. Since salt is not part of this particular database, the top suggestions provided by the methods, including garlic and various types of pepper, align with this requirement. Additionally, the recommendation to incorporate some form of fat, such as butter, bacon, or olive oil, align with culinary practices in various cuisines.

Similar to the observations in the soup database, the RBM method exhibits a higher probability of suggesting the top ingredients, whereas the Markov method and, to some extent, the NMF, present probabilities that are more evenly distributed among the suggested components, which emphasizes the RBM's confidence in proposing specific ingredients.

It is worth noting that although most of the recommendations align with common ingredients, as depicted in Fig.4.5.b), the methods actively avoided suggesting popular ingredients typically associated with sweet dishes, such as wheat, milk, cream, vanilla, and cane molasses, indicating a level of discernment in their recommendations.

4.5 Advantages and Disadvantages of the Proposed Methods

Considering the developed recipe generation and completion methods utilizing NMF, RBM, and Markov algorithms, it is of interest to conduct a comparative analysis. This evaluation aims to assess the performance of these methods across three different levels: very good, good and modest, encompassing a wide array of parameters, as illustrated in Fig.4.13.



	Performance		
Algorithm	Markov	RBM	NMF
Computational time complexity: Training	●●●	●●●	●
Computational time complexity: Suggesting	●	●●●	●●●
Ease of tuning	●●●	●	●●
Diversity of recipes	●	●●●	●●●
Maintaining ingredient correlations	●●	●●●	●●
Avoiding rare combinations	●	●●●	●●

Performance	
●●●	Very Good
●●	Good
●	Modest

Figure 4.13: Comparison of the relative performance of the developed methods (Markov, RBM and NMF) across a range of different parameters (Left), with corresponding symbols legend (Right).

Among the algorithms analyzed in Fig.4.13, the RBM consistently exhibits superior or comparable performance compared to both NMF and Markov algorithms across most parameters.

In terms of computational time complexity, the efficiency of the Markov and RBM algorithms in training stands out, particularly in extensive databases. Unlike the NMF, where it increases with $O(n_r^2)$, both RBM and Markov algorithms have a time complexity proportional only to n_r. However, concerning recommendation efficiency, both RBM and NMF algorithms surpass the Markov method, since the latter necessitates iterating through a substantial number of samples.

Additionally, concerning method tuning, Markov stands out for its simplicity, relying solely on ingredient pair frequencies without requiring parameter adjustments. In contrast, RBM proves more challenging, demanding tuning of multiple parameters: learning rate, number of hidden units and epochs, and batch size. The NMF falls in between, with the only parameter to determine being the number of features.

Furthermore, both the RBM and the NMF algorithms yield a diverse array of generated recipes, while the Markov method tends to repeatedly produce duplicated recipes due to its reliance on suggesting common ingredients.

Overall, the performance of the various methods in preserving the correlations and clusters within the databases was satisfactory, with the RBM notably outperforming the other two. However, some significant challenges were observed. The Markov algorithm exhibited remarkably strong correlations, while often failing to identify negative correlations. Meanwhile, the NMF algorithm tended to oversimplify the correlations, leading to the omission of smaller or weaker clusters.

Finally, when it came to handling prohibited ingredient pairs, the Markov algorithm was effective. Still, it struggled with separating uncommon combinations of popular ingredients, as these were prevalent in most generated recipes. On the other hand, the NMF algorithm satisfactorily separated unusual combinations of popular ingredients, but showed a higher number of recipes containing prohibited ingredient pairs compared to the other two methods. Notably, only the RBM demonstrated a strong performance in both aspects.

Chapter 5

Conclusion and Future Work

The rise of food allergies and obesity demands a transformation in the food industry, addressing individual needs. Simultaneously, the global struggle with hunger and food insecurity demands innovative solutions to enhance food systems and reduce waste. AI-enhanced recipes have the potential to offer personalization and to optimize production. Paired with smart food services, AI-enhanced recipes can provide cost-effective recipes and tailored nutrition, promoting accessible and healthy eating.

For this reason, three computational methods were used to complete and generate recipes: a non-Markovian chain, a restricted Boltzmann machine (RBM) and non-negative matrix factorization (NMF). The Markov based algorithm used, as transition matrix, the normalized frequencies of pairs of ingredients appearing together. What makes this chain non-Markovian is the fact that repetitions were not allowed, meaning that an ingredient could not be added twice to a recipe, and that not only the transition probabilities of one ingredient were considered, but the normalized sum of the transitions probabilities of all ingredients currently present in the incomplete recipe. The RBM, trained with contrastive divergence, employed 40 hidden units. The suggestions were made by passing the visible units (the incomplete recipes) through the RBM's hidden layer and then reconstructing the input data by passing the activations from the hidden layer back to the visible layer. Then, a new ingredient was chosen based on the probabilities originating from the normalization of the activations of the hidden layer. The NMF factorized the input matrix containing the recipe data into two matrices with 2 features. Then, to reconstruct the incomplete recipes, their corresponding vectors were multiplied by the pseudo-inverse of the input matrix and the two factorized matrices obtained.

Two metrics were used to characterize the recipes produced by the methods mentioned above and the databases employed, one comprising a collection of both clear and thick soups, and the second being a world cuisine database, featuring a wide array of recipes from diverse cultural backgrounds: mutual information and the Pearson correlation coefficient. A hierarchical clustering analysis was conducted using the Pearson correlation coefficient, considering the top 50 most common ingredients from the two databases. The hierarchical clustering process revealed clusters of ingredients that exhibited strong affinities with one another. Subsequently, 10000 recipes were generated for each algorithm trained on the respective databases.

In terms of computational time complexity, both the Markov and RBM methods demonstrated superior efficiency in training, especially for large databases, and the RBM and NMF methods outperformed Markov in providing faster suggestions. However, concerning parameter tuning, the Markov method stands out as the simplest, requiring no specific parameters. The NMF is relatively straightforward to tune, demanding only the determination of the optimal number of features representing the data. On the other hand, the RBM algorithm presents a greater challenge as it necessitates the selection of multiple parameters: number of hidden units, learning rate, number of epochs, and batch size.

The RBM and NMF algorithms produced highly diverse recipes, with the RBM generating 99.98% and 99.02% unique recipes when trained on the soup database and the world cuisine database, respectively, and the NMF generating 99.99% and 99.80% unique recipes when trained on the same databases. In contrast, the Markov algorithm displayed a tendency to generate repetitive recipes, generating only 29.33% and 36.21% of unique recipes when trained on the same databases.

Furthermore, the RBM demonstrated superior performance in preserving the original mutual information and ingredient correlations within the generated recipes compared to the source databases. Specifically, the Euclidean distances between the correlation coefficients were 4.06 and 4.25 for the soup and world cuisine databases, respectively. Additionally, for the weighted mutual information, the RBM achieved distances of 0.38 and 0.51 for the same databases. Additionally, both the Markov and NMF methods exhibited similar Euclidean distances for the correlation coefficients: 5.76 and 5.83, respectively, for the soup database, and 4.74 and 4.71, respectively, for the world cuisine database. However, NMF outperformed the Markov algorithm in terms of weighted mutual information, achieving distances of 0.52 and 0.49 compared to 0.91 and 1.04 obtained for the Markov algorithm.

When comparing the capabilities of the methods in avoiding prohibited pairs of ingredients, the Markov and the RBM outperformed the NMF. The Markov algorithm had a cumulative of 83 simulated recipes, out of the 20000, 10000 for each database, with one of the 8 prohibited pairs analyzed, while the RBM had 63, and the NMF had 586. However, for avoiding rare ingredient combinations of popular ingredients, the RBM and NMF outperformed the Markov. For example, in the soup database, while 8.30% of recipes contained chicken stock and water, the Markov method generated this combination in 81.52% of cases, while RBM and NMF limited it to 4.03% and 7.65%, respectively. Furthermore, in the world cuisine database, where 0.59% of recipes had onion and vanilla, the Markov method generated recipes with this combinations in

28.16% of cases, while the RBM and NMF did so in only 0.04% and 1.64%, respectively.

In the example recipe featuring onion, tomato, parsley, beef, and potato, all three methods trained on both databases performed well, consistently suggesting seasonings and liquids in the top 5 recommendations for the soup database. They also suggested seasonings and fats for the world cuisine database, while avoiding ingredients typically found in sweet dishes in their top 5 suggestions, indicating their ability to recognize ingredient categories. Furthermore, the RBM displayed higher confidence by having higher probabilities of selecting the top ingredients.

While the present study offers valuable insights in the area of completing and generating recipes, it is crucial to acknowledge its inherent limitations. One such limitation is the potential for biases within the training datasets, which can influence the discovered correlations. Additionally, the simplifications employed in representing ingredient interactions, though necessary for computational feasibility, may have resulted in the loss of certain subtleties.

About improvements and future work, focusing on correlations involving 3 or 4 ingredients holds the potential to deepen the understanding of how effectively the proposed methods capture the intricate relationships within recipes. One way to achieve this is through the Apriori algorithm, as it can identify frequent ingredient combinations and correlations, having the potential to be used to study these prevalent patterns and how other methods incorporate and respect them.

In addition, soliciting user feedback is crucial for assessing recipe satisfaction and coherence, since analytically quantifying effectiveness remains challenging due to the subjective nature of culinary experiences, making user input invaluable.

Addressing health concerns involves identifying allergens and suggesting healthier ingredient options. Furthermore, incorporating local and sustainable options, as well as catering to different culinary preferences, ensures inclusivity and sustainability.

Precise ingredient quantities, instead of binary options, would enhance the usability of recipe enhancement systems. Moreover, examining food groups in incomplete recipes presents an opportunity to offer more contextually relevant suggestions. By understanding what is lacking in the recipes, AI systems can provide targeted and meaningful enhancements.

Finally, integrating natural language processing techniques to comprehend and modify cooking steps alongside ingredient alterations could substantially enhance the adaptability and usability of AI-enhanced recipes. This integration would enable the system to not only understand the ingredients but also the cooking processes.

Bibliography

[1] M. Verma, R. Hontecillas, N. Tubau-Juni, V. Abedi, and J. Bassaganya-Riera, "Challenges in personalized nutrition and health," *Frontiers in Nutrition*, p. 117, 2018.

[2] A. Fiocchi, "Food allergy." `https://www.worldallergy.org/education-and-programs/education/allergic-disease-resource-center/professionals/food-allergy`, World Allergy Organization, 2017. Accessed: 15/08/2023.

[3] World Health Organization, "Obesity and overweight [fact sheet]." `https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight`, 2021. Accessed: 15/08/2023.

[4] FAO, IFAD, UNICEF, WFP and WHO, "The state of food security and nutrition in the world 2023. urbanization, agrifood systems transformation and healthy diets across the rural–urban continuum," 2023.

[5] FAO, "The state of food and agriculture 2019. moving forward on food loss and waste reduction.," 2019.

[6] United Nations Environment Programme, "Food waste index report 2021," *Nairobi*, 2021.

[7] M. Xu, J. M. David, S. H. Kim, *et al.*, "The fourth industrial revolution: Opportunities and challenges," *International journal of financial research*, vol. 9, no. 2, pp. 90–95, 2018.

[8] A. Hassoun, A. Aït-Kaddour, A. M. Abu-Mahfouz, N. B. Rathod, F. Bader, F. J. Barba, A. Biancolillo, J. Cropotova, C. M. Galanakis, A. R. Jambrak, *et al.*, "The fourth industrial revolution in the food industry—part i: Industry 4.0 technologies," *Critical Reviews in Food Science and Nutrition*, pp. 1–17, 2022.

[9] A. Hassoun, A. E.-D. Bekhit, A. R. Jambrak, J. M. Regenstein, F. Chemat, J. D. Morton, M. Gudjónsdóttir, M. Carpena, M. A. Prieto, P. Varela, *et al.*, "The fourth industrial revolution in the food industry—part ii: Emerging food trends," *Critical Reviews in Food Science and Nutrition*, pp. 1–31, 2022.

[10] United Nations, "Sustainable development goals." `https://www.un.org/sustainabledevelopment`, 2013. Accessed: 13/08/2023.

[11] A. Lucena, A. S. Freitas, A. L. Ferreira, and F. Abreu, "Inspiration from systematic literature reviews to predict the future of food services in smart cities," in *SMART 2023, The Twelfth International Conference on Smart Cities, Systems, Devices and Technologies*, ThinkMind, IARIA, June 2023.

[12] S. M. Noble, M. Mende, D. Grewal, and A. Parasuraman, "The fifth industrial revolution: How harmonious human–machine collaboration is triggering a retail and service [r] evolution," *Journal of Retailing*, vol. 98, no. 2, pp. 199–208, 2022.

[13] G. Y. Kim and J.-S. Seo, "A new paradigm for clinical nutrition services in the era of the fourth industrial revolution," *Clinical Nutrition Research*, vol. 10, no. 2, p. 95, 2021.

[14] Y.-Y. Ahn, S. E. Ahnert, J. P. Bagrow, and A.-L. Barabási, "Flavor network and the principles of food pairing," *Scientific reports*, vol. 1, no. 1, p. 196, 2011.

[15] F. Andres, "Data engineering challenges in intelligent food and cooking recipes," in *2023 IEEE 39th International Conference on Data Engineering Workshops (ICDEW)*, pp. 214–217, IEEE, 2023.

[16] M. Goel and G. Bagler, "Computational gastronomy: A data science approach to food," *Journal of Biosciences*, vol. 47, no. 1, p. 12, 2022.

[17] P. Hamet and J. Tremblay, "Artificial intelligence in medicine," *Metabolism*, vol. 69, pp. S36–S40, 2017.

[18] J. McCarthy *et al.*, "What is artificial intelligence," 2007.

[19] S. J. Russell, *Artificial intelligence a modern approach.* Pearson Education, Inc., 2010.

[20] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.

[21] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[22] N. Sebe, *Machine learning in computer vision*, vol. 29. Springer Science & Business Media, 2005.

[23] L. M. Policarpo, D. E. da Silveira, R. da Rosa Righi, R. A. Stoffel, C. A. da Costa, J. L. V. Barbosa, R. Scorsatto, and T. Arcot, "Machine learning through the lens of e-commerce initiatives: An up-to-date systematic literature review," *Computer Science Review*, vol. 41, p. 100414, 2021.

[24] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 2, pp. 604–624, 2020.

[25] A. Rajkomar, J. Dean, and I. Kohane, "Machine learning in medicine," *New England Journal of Medicine*, vol. 380, no. 14, pp. 1347–1358, 2019.

[26] I. El Naqa and M. J. Murphy, *What is machine learning?* Springer, 2015.

[27] S. G. Shaikh, B. Suresh Kumar, and G. Narang, "Recommender system for health care analysis using machine learning technique: A review," *Theoretical Issues in Ergonomics Science*, vol. 23, no. 5, pp. 613–642, 2022.

[28] M. Alloghani, D. Al-Jumeily, J. Mustafina, A. Hussain, and A. J. Aljaaf, "A systematic review on supervised and unsupervised machine learning algorithms for data science," *Supervised and unsupervised learning for data science*, pp. 3–21, 2020.

[29] A. Borghini, "What is a recipe?," *Journal of Agricultural and Environmental Ethics*, vol. 28, pp. 719–738, 2015.

[30] A. Salvador, M. Drozdzal, X. Giró-i Nieto, and A. Romero, "Inverse cooking: Recipe generation from food images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10453–10462, 2019.

[31] J. Chen, B. Zhu, C.-W. Ngo, T.-S. Chua, and Y.-G. Jiang, "A study of multi-task and region-wise deep learning for food ingredient recognition," *IEEE Transactions on Image Processing*, vol. 30, pp. 1514–1526, 2020.

[32] M. Nonaka, K. Otake, and T. Namatame, "Evaluation of cooking recipes using their texts and images," in *International Conference on Human-Computer Interaction*, pp. 312–322, Springer, 2021.

[33] Y. Sugiyama and K. Yanai, "Cross-modal recipe embeddings by disentangling recipe contents and dish styles," in *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 2501–2509, 2021.

[34] Z. Xie, L. Liu, L. Li, and L. Zhong, "Learning joint embedding with modality alignments for cross-modal retrieval of recipes and food images," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 2221–2230, 2021.

[35] H. Wang, G. Lin, S. C. Hoi, and C. Miao, "Decomposing generation networks with structure prediction for recipe generation," *Pattern Recognition*, vol. 126, p. 108578, 2022.

[36] C. Trattner and D. Elsweiler, "Food recommender systems: important contributions, challenges and future research directions," *arXiv preprint arXiv:1711.02760*, 2017.

[37] R. Yera, A. A. Alzahrani, L. Martínez, and R. M. Rodríguez, "A systematic review on food recommender systems for diabetic patients," *International Journal of Environmental Research and Public Health*, vol. 20, no. 5, p. 4248, 2023.

[38] R. Y. Toledo, A. A. Alzahrani, and L. Martinez, "A food recommender system considering nutritional information and user preferences," *IEEE Access*, vol. 7, pp. 96695–96711, 2019.

[39] P. Chavan, B. Thoms, and J. Isaacs, "A recommender system for healthy food choices: building a hybrid model for recipe recommendations using big data sets," *Hawaii International Conference on System Sciences*, 2021.

[40] M. Rostami, V. Farrahi, S. Ahmadian, S. M. J. Jalali, and M. Oussalah, "A novel healthy and time-aware food recommender system using attributed community detection," *Expert Systems with Applications*, vol. 221, p. 119719, 2023.

[41] D. Park, K. Kim, Y. Park, J. Shin, and J. Kang, "Kitchenette: Predicting and recommending food ingredient pairings using siamese neural networks," *arXiv preprint arXiv:1905.07261*, 2019.

[42] D. Park, K. Kim, S. Kim, M. Spranger, and J. Kang, "Flavorgraph: a large-scale food-chemical graph for generating food representations and recommending food pairings," *Scientific reports*, vol. 11, no. 1, p. 931, 2021.

[43] B. P. Majumder, S. Li, J. Ni, and J. McAuley, "Generating personalized recipes from historical user preferences," *arXiv preprint arXiv:1909.00105*, 2019.

[44] H. H. Lee, K. Shu, P. Achananuparp, P. K. Prasetyo, Y. Liu, E.-P. Lim, and L. R. Varshney, "Recipegpt: Generative pre-training based cooking recipe generation and evaluation system," in *Companion Proceedings of the Web Conference 2020*, pp. 181–184, 2020.

[45] J. Fujita, M. Sato, and H. Nobuhara, "Model for cooking recipe generation using reinforcement learning," in *2021 IEEE 37th International Conference on Data Engineering Workshops (ICDEW)*, pp. 1–4, IEEE, 2021.

[46] M. Gim, D. Park, M. Spranger, K. Maruyama, and J. Kang, "Recipebowl: A cooking recommender for ingredients and recipes using set transformer," *IEEE Access*, vol. 9, pp. 143623–143633, 2021.

[47] M. Gim, D. Choi, K. Maruyama, J. Choi, H. Kim, D. Park, and J. Kang, "Recipemind: Guiding ingredient choices from food pairing to recipe completion using cascaded set transformer," in *Proceedings of the 31st ACM international conference on information & knowledge management*, pp. 3092–3102, 2022.

[48] P. F. Cueto, M. Roet, and A. Słowik, "Completing partial recipes using item-based collaborative filtering to recommend ingredients," *arXiv preprint arXiv:1907.12380*, 2019.

[49] M. De Clercq, M. Stock, B. De Baets, and W. Waegeman, "Data-driven recipe completion using machine learning methods," *Trends in Food Science & Technology*, vol. 49, pp. 1–13, 2016.

[50] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: Techniques, applications, and challenges," *Recommender Systems Handbook*, pp. 1–35, 2021.

[51] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[52] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295, 2001.

[53] J. Wang, A. P. De Vries, and M. J. Reinders, "Unifying user-based and item-based collaborative filtering approaches by similarity fusion," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 501–508, 2006.

[54] R. R. Kumar, G. Apparao, and S. Anuradha, "Deep scalable and distributed restricted boltzmann machine for recommendations," *International Journal of System Assurance Engineering and Management*, pp. 1–13, 2022.

[55] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the 24th international conference on Machine learning*, pp. 791–798, 2007.

[56] C. J. Geyer, "Introduction to markov chain monte carlo," *Handbook of markov chain monte carlo*, vol. 20116022, 2011.

[57] A. A. Markov, "Extension of the law of large numbers to dependent quantities," *Izv. Fiz.-Matem. Obsch. Kazan Univ.(2nd Ser)*, vol. 15, no. 1, pp. 135–156, 1906.

[58] E. Seneta, "Markov and the creation of markov chains," in *Markov Anniversary Meeting*, pp. 1–20, Citeseer, 2006.

[59] N. G. Van Kampen, *Stochastic processes in physics and chemistry*, vol. 1. Elsevier, 1992.

[60] J. R. Norris, *Markov chains.* No. 2, Cambridge university press, 1998.

[61] G. Bhanot, "The metropolis algorithm," *Reports on Progress in Physics*, vol. 51, no. 3, 1988.

[62] L. Pretto, "A theoretical analysis of google's pagerank," in *String Processing and Information Retrieval: 9th International Symposium, SPIRE 2002 Lisbon, Portugal, September 11–13, 2002 Proceedings 9*, pp. 131–144, Springer, 2002.

[63] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer networks and ISDN systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[64] S. Chakrabarti, *Mining the Web: Discovering knowledge from hypertext data.* Morgan Kaufmann, 2002.

[65] D. Vise, "The google story," *Strategic Direction*, vol. 23, no. 10, 2007.

[66] T. Seymour, D. Frantsvog, S. Kumar, *et al.*, "History of search engines," *International Journal of Management & Information Systems (IJMIS)*, vol. 15, no. 4, pp. 47–58, 2011.

[67] T. L. Griffiths, M. Steyvers, and A. Firl, “Google and the mind: Predicting fluency with pagerank,” *Psychological science*, vol. 18, no. 12, pp. 1069–1076, 2007.

[68] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, *Handbook of markov chain monte carlo.* CRC press, 2011.

[69] M. Quadrana, P. Cremonesi, and D. Jannach, “Sequence-aware recommender systems,” *ACM computing surveys (CSUR)*, vol. 51, no. 4, pp. 1–36, 2018.

[70] S. Raza and C. Ding, “Progress in context-aware recommender systems—an overview,” *Computer Science Review*, vol. 31, pp. 84–97, 2019.

[71] E. S. Khorasani, Z. Zhenge, and J. Champaign, “A markov chain collaborative filtering model for course enrollment recommendations,” in *2016 IEEE International Conference on Big Data (Big Data)*, pp. 3484–3490, IEEE, 2016.

[72] M. H. Aghdam, “Context-aware recommender systems using hierarchical hidden markov model,” *Physica A: Statistical Mechanics and Its Applications*, vol. 518, pp. 89–98, 2019.

[73] P. Smolensky, “Information processing in dynamical systems: Foundations of harmony theory,” tech. rep., Colorado Univ at Boulder Dept of Computer Science, 1986.

[74] V. Upadhya and P. Sastry, “An overview of restricted boltzmann machines,” *Journal of the Indian Institute of Science*, vol. 99, no. 2, pp. 225–236, 2019.

[75] H. Larochelle, M. Mandel, R. Pascanu, and Y. Bengio, “Learning algorithms for the classification restricted boltzmann machine,” *The Journal of Machine Learning Research*, vol. 13, pp. 643–669, 2012.

[76] A. Fischer and C. Igel, “An introduction to restricted boltzmann machines,” in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 17th Iberoamerican Congress, CIARP 2012, Buenos Aires, Argentina, September 3-6, 2012. Proceedings 17*, pp. 14–36, Springer, 2012.

[77] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[78] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, *et al.*, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[79] N. Zhang, S. Ding, J. Zhang, and Y. Xue, “An overview on restricted boltzmann machines,” *Neurocomputing*, vol. 275, pp. 1186–1199, 2018.

[80] G. E. Hinton, “A practical guide to training restricted boltzmann machines,” *Neural Networks: Tricks of the Trade: Second Edition*, pp. 599–619, 2012.

[81] D. K. Behera, M. Das, and S. Swetanisha, “Predicting users’ preferences for movie recommender system using restricted boltzmann machine,” in *Computational Intelligence in Data Mining: Proceedings of the International Conference on CIDM 2017*, pp. 759–769, Springer, 2019.

[82] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, “An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems,” *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1273–1284, 2014.

[83] S. Fathi Hafshejani and Z. Moaberfard, “Initialization for non-negative matrix factorization: a comprehensive review,” *International Journal of Data Science and Analytics*, vol. 16, no. 1, pp. 119–134, 2023.

[84] G. W. Stewart, “On the early history of the singular value decomposition,” *SIAM review*, vol. 35, no. 4, pp. 551–566, 1993.

[85] R. Agrawal, R. Srikant, *et al.*, “Fast algorithms for mining association rules,” in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, pp. 487–499, Santiago, Chile, 1994.

[86] P. Weiderer, A. M. Tomé, and E. W. Lang, “A nmf-based extraction of physically meaningful components from sensory data of metal casting processes,” *Journal of Manufacturing Systems*, vol. 54, pp. 62–73, 2020.

[87] Z.-Y. Zhang, “Nonnegative matrix factorization: models, algorithms and applications,” *Data Mining: Foundations and Intelligent Paradigms: Volume 2: Statistical, Bayesian, Time Series and other Theoretical Aspects*, pp. 99–134, 2012.

[88] H. Gao, F. Nie, W. Cai, and H. Huang, “Robust capped norm nonnegative matrix factorization: Capped norm nmf,” in *Proceedings of the 24th ACM international on conference on information and knowledge management*, pp. 871–880, 2015.

[89] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[90] Z. Khan, N. Iltaf, H. Afzal, and H. Abbas, “Enriching non-negative matrix factorization with contextual embeddings for recommender systems,” *Neurocomputing*, vol. 380, pp. 246–258, 2020.

[91] R. Ievgen and B. Younés, “Random subspaces nmf for unsupervised transfer learning,” in *2014 International Joint Conference on Neural Networks (IJCNN)*, pp. 3901–3908, IEEE, 2014.

[92] C. Graham, *Markov chains: analytic and Monte Carlo computations.* John Wiley & Sons, 2014.

[93] N. Madras and G. Slade, *The self-avoiding walk.* Springer Science & Business Media, 2013.

[94] P. D. Hoff, *A first course in Bayesian statistical methods*, vol. 580. Springer, 2009.

[95] G. E. Hinton, “Training products of experts by minimizing contrastive divergence,” *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[96] M. A. Nielsen, *Neural networks and deep learning*, vol. 25. Determination press San Francisco, CA, USA, 2015.

[97] L. Batina, B. Gierlichs, E. Prouff, M. Rivain, F.-X. Standaert, and N. Veyrat-Charvillon, “Mutual information analysis: a comprehensive study,” *Journal of Cryptology*, vol. 24, no. 2, pp. 269–291, 2011.

[98] A. Kraskov, H. Stögbauer, and P. Grassberger, “Estimating mutual information,” *Physical review E*, vol. 69, no. 6, p. 066138, 2004.

[99] R. S. Witte and J. S. Witte, *Statistics.* John Wiley & Sons, 2017.

[100] B. L. Agarwal, *Basic statistics.* New Age International, 2006.

[101] F. Nielsen, *Introduction to HPC with MPI for Data Science.* Springer, 2016.

[102] X. Liu, X.-H. Zhu, P. Qiu, and W. Chen, “A correlation-matrix-based hierarchical clustering method for functional connectivity analysis,” *Journal of neuroscience methods*, vol. 211, no. 1, pp. 94–102, 2012.

[103] S. Miyamoto, R. Abe, Y. Endo, and J.-I. Takeshita, “Ward method of hierarchical clustering for non-euclidean similarity measures,” in *2015 7th International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, pp. 60–63, IEEE, 2015.

[104] C. Forbes, M. Evans, N. Hastings, and B. Peacock, *Statistical distributions.* John Wiley & Sons, 2011.

[105] T. Mallick, M. Kiran, B. Mohammed, and P. Balaprakash, “Dynamic graph neural network for traffic forecasting in wide area networks,” in *2020 IEEE International Conference on Big Data (Big Data)*, pp. 1–10, IEEE, 2020.

[106] L. Wang, Y. Zhang, and J. Feng, “On the euclidean distance of images,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1334–1339, 2005.