RecipeRAG: A Knowledge Graph-Driven Approach to Personalized Recipe Retrieval and Generation

Julie Loesch[1], Emin Durmuş[2] and Remzi Celebi[1]

[1]Department of Advanced Computing Sciences, Maastricht University, Paul-Henri Spaaklaan 1, Maastricht, 6229 EN, Netherlands
[2]Kirklareli University

Abstract

Recommending or generating a recipe that satisfies the growing diversity of dietary needs and preferences is a significant challenge for many people. Food choices are influenced by a mix of factors (taste, dietary restriction, health, availability of ingredients), yet many algorithms are optimized for single-criterion decisions. In this work, we introduce RecipeRAG, a novel knowledge graph based retrieval-augmented generation system for personalized recipe generation based on users' multiple criteria. We construct RecipeKG, a knowledge graph derived from Food.com and enriched with additional semantic tags, to capture complex relationships between recipes and their related concepts and use knowledge graph embedding models for retrieval. RecipeRAG employs multi-criteria information retrieval on RecipeKG on user-defined constraints, followed by a large language model to generate personalized recipes. Our experiments demonstrate that RecipeRAG outperforms existing methods in both retrieval and generation tasks, producing high-quality personalized recipes that meet multiple constraints. RecipeRAG offers a promising solution to make a connection between traditional recipes and evolving nutritional and dietary needs, allowing for more flexible and personalized cooking options.

Keywords

Knowledge Graph (KG), Large Language Model (LLM), Retrieval-Augmented Generation (RAG)

1. Introduction

Designing or adapting recipes under multiple constraints—such as dietary needs, taste, nutrition, availability, and cultural norms—is a complex multi-objective combinatorial problem. Traditional approaches, including rule-based systems and single-objective optimization, struggle to handle these competing goals simultaneously [1]. Recent models like GISMo [2] and SHARE [3] support ingredient substitution and recipe editing under dietary constraints. However, they face key limitations: low substitution accuracy, limited modeling of taste and function, lack of multi-objective optimization, and minimal user interaction or explainability.

To address these gaps, intelligent recipe generation systems must move beyond surface-level keyword matching to reason about the underlying semantics of food. This requires a deep understanding of ingredient properties, culinary techniques, and their interrelationships. Food ontologies provide this foundation by capturing rich, structured knowledge on nutrition, ingredient hierarchies, cooking methods, and safety considerations [4]. Leveraging such ontologies enables more informed, context-aware recipe recommendations that better satisfy complex, user-defined constraints.

Recent research explores the potential of generating recipes based on specific constraints, enabling systems to create dishes that align with user requirements. However, most existing studies optimize for a *single constraint at a time*. For instance, Kazama et al. [5] focus on regional cuisine styles, while Shirai et al. [6] develop models that generate nutritionally optimized recipes. Morales-Garzón et al. [7] concentrate on vegetarian recipe generation. Li et al. [3] propose a system for editing recipes to accommodate dietary restrictions; however, their approach considers only recipe titles and ingredients,

RAGE-KG 2025: The Second International Workshop on Retrieval-Augmented Generation Enabled by Knowledge Graphs, co-located with ISWC 2025, November 2–6, 2025, Nara, Japan
julie.loesch@maastrichtuniversity.nl (J. Loesch); emin.durmuss80@gmail.com (E. Durmuş); remzi.celebi@maastrichtuniversity.nl (R. Celebi)
0000-0001-7769-4272 (R. Celebi)

neglecting cooking instructions, which are critical for modeling preparation logic and ingredient interactions.

Beyond these limitations, recent LLM-based models introduce new challenges. They often hallucinate unsafe or impractical recipes, such as incompatible ingredient substitutions or incomplete cooking procedures. Furthermore, many systems lack the ability to handle multiple, user-defined constraints simultaneously, and often operate without integrating structured knowledge like food ontologies. This results in limited adaptability, poor interpretability, and unreliable outcomes, especially when recipes must align with complex health or cultural considerations.

Retrieval-Augmented Generation (RAG) systems can assist in generating recipes that meet specific constraints while considering the full recipe context. These systems enhance text generation by integrating external knowledge. Given a user's input, RAG models retrieve the top k relevant knowledge pieces to inform the generation process. While basic RAG systems typically rely on textual data, other sources, such as ontologies or knowledge graphs, can also serve as their knowledge base. The use of knowledge graph-based RAG systems has recently gained increasing attention.

Consequently, this paper introduces a knowledge graph-based retrieval-augmented generation framework for recipe generation, called *RecipeRAG*. The proposed approach enables users to generate recipes while incorporating constraints. The contributions of our work are as follows:

- A new recipe knowledge graph, *RecipeKG*, is constructed from Food.com.
- A retrieval-augmented generation system is developed that leverages a knowledge graph to generate personalized recipes.
- Instead of relying solely on direct similarity matching between user queries and recipes, RecipeRAG introduces a novel retrieval method that frames recipe recommendation as a multi-criteria link prediction task, enabling the system to optimize across user-defined constraints.

2. Related Work

Various approaches have been proposed for recipe generation. In this section, three distinct subsets of recipe generation methods will be examined.

2.1. Generating Cooking Instructions

Instruction generation for cooking recipes has advanced significantly with the integration of large language models (LLMs) and traditional language models. These methods focus on producing coherent, structured, and detailed cooking instructions.

Fine-tuned GPT-2 models have demonstrated strong capabilities in this area. RecipeGPT [8] supports both ingredient and instruction generation, while Goel et al. [9] trained GPT-2 on the RecipeDB dataset to generate novel recipe instructions. Similarly, Hwang et al. [10] employed a prompt-based framework with LLMs to simplify recipe steps while retaining essential details.

More recent advancements include multimodal approaches. LLaVa-Chef [11] fine-tunes the LLaVA model to integrate visual and textual data, enhancing instruction accuracy and outperforming existing models like GPT-2 and LLaMA in instruction quality.

Beyond LLMs, traditional language models continue to contribute. Liu et al. [12] proposed a counterfactual recipe generation method that modifies a base recipe based on a change in an ingredient, while logically adjusting the subsequent steps. Transformer-based models have also gained traction. Majumder et al. [13] expanded recipe instructions from incomplete ingredient lists, while Liu et al. [14] introduced a structured, step-by-step planning approach for refining generated instructions.

Li et al. [3] combined recipe editing with instruction generation, using a copy attention mechanism to align cooking steps with modified ingredient lists.

2.2. Generating Ingredient Selection

Ingredient selection plays a crucial role in shaping the flavor profile and structure of a recipe. Computational methods have been developed to suggest novel ingredient combinations, leveraging knowledge graphs and embedding models.

Pini et al. [15] constructed a knowledge graph from food databases to recommend new ingredient combinations based on similarity metrics. By capturing ingredient features such as flavor profiles and functional attributes, their tool generates ranked ingredient suggestions tailored to user-defined constraints.

Embedding models have also been employed for ingredient substitutions. Chen et al. [16] introduced NutRec, a framework that modifies ingredient lists to create healthier recipes. NutRec first predicts potential ingredients that could be added to the original ingredient list using an embedding-based model and second estimates optimal ingredient quantities with a neural network, ensuring nutritional balance.

Similarly, Pan et al. [17] proposed a two-step method for generating recipes with novel ingredients. They vectorized ingredients using Doc2Vec, identified substitutions based on cosine similarity, and generated coherent recipes using N-gram and LSTM-based models.

2.3. Constraint-Based Generation

Generating recipes that align with user preferences (i.e., dietary restrictions, ingredient availability, or nutritional needs) is a complex challenge. Various approaches leverage recipe embeddings, knowledge graphs, and transformer-based models to address this.

Kazama et al. [5] introduced a framework that converts traditional recipes into regional variations. This system includes a neural network that calculates the contribution of each ingredient to specific regional cuisines and an extended Word2Vec model that recommends new ingredients matching the target regional style while maintaining high similarity to the original recipe.

Similarly, Morales-Garzón et al. [7] proposed an unsupervised recipe editing method using Word2Vec embeddings trained on cooking texts. Ingredients from original recipes are mapped to a food composition database, and unsuitable ingredients are replaced with alternatives that meet specified constraints.

Shirai et al. [6] developed a constraint-based ingredient substitution method using FoodKG, leveraging knowledge graphs to capture rich semantic relationships.

Li et al. [3] introduced SHARE (System for Hierarchical Assistive Recipe Editing), a transformer-based model for recipe editing under dietary constraints. SHARE uses two encoder-decoder networks: the first replaces ingredients based on user constraints, and the second generates new cooking instructions tailored to the modified recipe.

3. Data

This section describes the construction of the new knowledge graph from Food.com, called RecipeKG, as well as the generation of the ground truth datasets used to evaluate the retrieval and generation modules of RecipeRAG.

3.1. RecipeKG (Training Set)

The dataset used in this paper is sourced from *Food.com-Recipes and Reviews*[1]. We removed recipes containing null values from the original dataset. For the remaining $88{,}519$ recipes, we generated labels based on nutritional information and user-generated descriptions, categorizing them into 6 distinct categories. The description of each category is presented in Table 1.

However, the dataset, which contains $88{,}519$ recipes, exhibits highly imbalanced label distributions across categories.To balance the dataset, we grouped some of the underrepresented labels based on

[1]https://www.kaggle.com/datasets/irkaal/foodcom-recipes-and-reviews

Table 1
The description of each category associated with a given recipe.

Category	Description	Examples
Cook Time	The time the user needs to cook for a recipe.	Less than 30 mins
Recipe Servings	The number of people a recipe serves.	1-4 servings
Diet Type	The type of diet a recipe attempts to achieve.	Lactose Free, Vegetarian
Healthy Type	The nutrient evaluation of a recipe.	High Protein, High Fat
Region	The cuisine style of a recipe.	Asia
Meal Type	The type of meal a recipe belongs to.	Breakfast, Lunch

similar attributes. After reducing the number of samples from overrepresented classes, a total of 6,754 recipes were used to construct RecipeKG.

3.2. Dataset to Evaluate Multi-criteria Retrieval (Retrieval Test Set 1)

To assess the retrieval performance of RecipeRAG, we created five separate ground truth subsets from our collection of 6,754 recipes to evaluate different levels of user query complexity. Each subset represents a specific number of combined user criteria, ranging from 1 to 5, and contains 100 unique combinations of criteria. An example of a 2 criteria set is `(hasDietType = Vegetarian and isFromRegion = Asian)`, which means that only two criteria are taken into account simultaneously. This allows us to measure how the system performs on various queries, from simple to more complex queries.

To generate the combination sets, we used all possible options for each criterion, as the number of options was relatively small, ranging from 3 to 10. For example, `hasCarbLevel` can take three values `(low_carb, medium_carb, high_carb)`, while `Region` can have up to 10 options `(africa, asia, europe, global, indian, latin_america_and_caribbean, mediterranean, middle_east, north_america, oceania)`.

For smaller criteria combinations (1 and 2), we exhaustively enumerated all possible combinations and retained those for which at least one matching recipe was found. For larger set sizes (3 to 5), where the combinatorial space increases exponentially, we employed a stratified sampling strategy to ensure both feasibility and representativeness. The procedure was as follows:

1. All possible combinations were generated and filtered to retain only those with at least one corresponding recipe.
2. We computed dynamic bin sizes using the Freedman–Diaconis rule [18], which determines the optimal bin width based on the interquartile range (IQR) and sample size.
3. The combinations were then grouped into quantile-based bins according to their match count distributions.
4. For each bin, we randomly selected 50% of the data.

This methodology produced a balanced set, referred to as Retrieval Test Set 1, which includes both simple single-criterion queries and more complex multi-criteria combinations. The total number of possible and sampled combinations for Retrieval Test Set 1 is presented in Table 3. Table 2 provides an overview of all criteria and their corresponding values considered in this study.

3.3. Dataset to Evaluate Multi-ingredient Retrieval (Retrieval Test Set 2)

In addition, we developed a dedicated test set focusing exclusively on ingredient-based recipe retrieval. This test set aims to assess the system's performance on a common real-world task: retrieving recipes that contain specific combinations of ingredients.

We began by conducting a comprehensive analysis of the ingredient distribution within the dataset. From a total of 88,519 recipes, we identified 1,646 unique ingredients, exhibiting a characteristic long-tail distribution. A small subset of staple ingredients (i.e., salt, butter, sugar) appeared in thousands of recipes, while the majority of specialty ingredients occurred only infrequently.

Table 2
Overview of criteria applied in this study.

Attribute	Values
hasProteinLevel	Low Protein, Medium Protein, High Protein
hasCarbLevel	Low Carb, Medium Carb, High Carb
hasFatLevel	Low Fat, Medium Fat, High Fat
hasCalorieLevel	Low Calorie, Medium Calorie, High Calorie
isForMealType	breakfast, lunch, dinner, snack, dessert, starter, brunch, drink
hasDietType	Vegetarian, Vegan, Paleo, Standard
hasCuisineRegion	North America, Global, Mediterranean Europe, Northern/Western Europe, Latin America, East Asia, South Asia, Southeast Asia, Middle East & Anatolia, Oceania, Eastern Europe & Eurasia, Caribbean, Sub-Saharan Africa
usesCookingMethod	oven, pot, pan, no cook, barbecue, air fryer, microwave

To construct a representative yet computationally manageable sample of ingredients, we adopted the following methodology:

1. We generated a frequency distribution of all ingredients and confirmed a power-law distribution via log-log plotting.
2. We applied equal-frequency (quantile) binning, using a dynamically determined bin count of 16, computed via the Freedman–Diaconis rule to accommodate data dispersion and sample size.
3. Within each bin, we performed a stratified sampling of 25%, resulting in a total of 412 representative ingredients covering the full frequency spectrum, from ubiquitous staples to rare ingredients.
4. An inverted index was then constructed, mapping each selected ingredient to the corresponding set of recipes, enabling efficient intersection-based queries.

Using these 412 representative ingredients, we generated combinations of sizes 1 through 4 and evaluated their retrieval potential:

1. All possible combinations were computed using the binomial coefficient.
2. For each combination, the set of matching recipes was determined via intersection operations on the inverted index.
3. Only combinations with at least one matching recipe were retained for evaluation.

Table 3 presents the number of all possible and retained (sampled) ingredient combinations for Retrieval Test Set 2.

This systematic approach enables rigorous evaluation of ingredient-based retrieval capabilities, encompassing a wide range of query complexities from single-ingredient lookups to intricate multi-ingredient constraints.

Table 3
Number of combinations for Retrieval Test Set 1 (criteria-based) and Test Set 2 (ingredient-based). The set size denotes the number of criteria or ingredients (i.e., a set size of 2 corresponds to a pair of criteria or ingredients).

Set Size	Retrieval Test Set 1 (Criteria)		Retrieval Test Set 2 (Ingredients)	
	All Combinations	Sampled Combinations	All Combinations	Sampled Combinations
1	44	44	412	412
2	801	793	84,666	9,096
3	7,924	3,671	11,571,020	33,133
4	46,835	18,091	1,183,136,795	47,692
5	170,268	46,616	–	–

3.4. Dataset to Evaluate Recipe Generation (Generation Test Set)

To evaluate the generation capabilities of RecipeRAG, we randomly selected criteria from either Retrieval Test Set 1 or Retrieval Test Set 2. Subsequently, a random integer between 2 and 5 was drawn to determine the number of criteria to include. For Retrieval Test Set 2, which consists solely of ingredients and is relatively large, we focused on frequently occurring ingredients. Ingredient frequencies were calculated and categorized into low, medium, and high-frequency groups, with the selection limited to those in the high-frequency category.

4. Methodology

This section details the KG-based RAG system we developed to generate personalized recipes based on given user criteria. Figure 1 illustrates the overall structure and workflow of RecipeRAG.

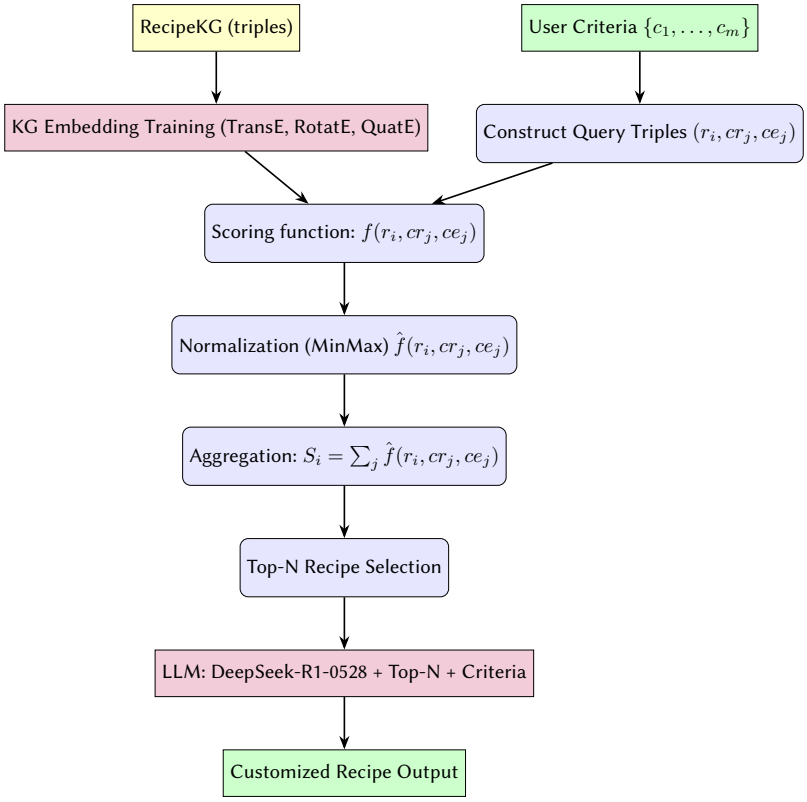

Figure 1: An overview of RecipeRAG, emphasizing the retrieval module.

RecipeRAG is composed of two main components: a knowledge graph-based retrieval module and a recipe generation module. Figure 1 highlights the retrieval module, which uses RecipeKG (depicted in Figure 2), a knowledge graph constructed from Food.com recipe data. This knowledge graph represents both recipes and user criteria as entities, along with the relationships between them.

Unlike traditional RAG systems that leverage direct graph structure, RecipeRAG employs a KG embedding-based scoring mechanism to retrieve relevant recipes. This process consists of following steps:

1. First, KG embedding models (i.e., TransE [19], RotatE [20] and QuatE [21]) were trained on RecipeKG to learn the embeddings of recipes, user criteria, and their relationships.
2. The system then applies multi-criteria optimization to calculate a plausibility score for each recipe with respect to each criterion. This score is derived from the embedding model's scoring function, reflecting how strongly a given connection is plausible.

Figure 2: Knowledge graph schema of RecipeKG.

Let:

$R = \{r_1, r_2, \ldots, r_n\}$ be the set of candidate recipes,

$C = \{c_1, c_2, \ldots, c_m\}$ be the set of user-specified criteria,

$f(r_i, c_j^r, c_j^e)$ be the *scoring function* from a trained KG embedding model which evaluates the plausibility of a triple (r_i, c_j^r, c_j^e), where r_i is a recipe, c_j^r is the relation for criterion c_j, and c_j^e is the entity for criterion c_j.

3. Each score for a criterion is then normalized to the $[0, 1]$ range using MinMaxScaler to ensure that the scores are on the same scale.
4. After normalization, the scores for each criterion are finally aggregated by recipe ID and then summed as follows:

The *aggregated score* for recipe r_i is:

$$S_i = \sum_{j=1}^{m} \hat{f}(r_i, c_j^r, c_j^e),$$

where $\hat{f}(r_i, c_j^r, c_j^e)$is the normalized score using MinMax scaling across recipes for each criterion c_j.

This aggregation produces a single overall score for each recipe that reflects how well a recipe meets all the user's specified criteria. This prevents any single criterion from dominating and ensures that multiple criteria are evaluated in a balanced way. Taking into account the contribution of each criterion, the scoring mechanism allows us to select the most appropriate recipes even in cases of incomplete or conflicting labels. Instead of relying solely on exact matches, the aggregated score offers a flexible and generalizable evaluation that considers all aspects of the user's input.

5. The retrieval process then selects the top-ranking recipes based on the aggregated scores. After retrieving the top N (N=5 used in the experiments) most relevant recipes, the system utilizes *DeepSeek-R1-0528* to generate new recipes. This model incorporates the retrieved recipes as context, along with the original user criteria, to produce customized recipe suggestions. If an insufficient number of recipes were retrieved, a fallback mechanism was applied.

5. Experiments

This section provides an overview of the experimental design used to evaluate the retrieval performance of RecipeRAG with different knowledge graph embedding models, as well as the generation performance for recipe recommendation and creation.

5.1. Knowledge Graph Embedding Evaluation

We trained three knowledge graph embedding (KGE) models to learn the vector representations of entities and relations within our knowledge graph:

1. **TransE:** A translational distance model that interprets relations as translations in the embedding space.
2. **RotatE:** A rotational model that represents relations through rotations in a complex vector space.
3. **QuatE:** A quaternion-based model that utilizes quaternions to encode entities and relations.

These models were implemented using the PyKEEN framework, a Python library designed for training and evaluating KGE models [22]. The training was conducted on our recipe knowledge graph, RecipeKG, which encompasses entities such as recipes, ingredients, diet types, and meal types, along with their corresponding relations.

Since text embedding is a widely used method in traditional RAG systems, we incorporated two text embedding baselines for comparison. The first baseline, *GTE-Large*, is a general-purpose text embedding model trained with multi-stage contrastive [23]. The second baseline, *all-MiniLM-L6-v2* (MiniLM) from sentence-transformers, has been optimized for clustering and semantic search tasks. These models were chosen for their effectiveness across various applications: GTE-large represents advancements in text embedding techniques, while MiniLM is well-suited for semantic search, aligning closely with our recipe retrieval objectives.

To evaluate retrieval performance, we used the two test datasets described in Subsections 3.2 and 3.3. We then evaluated the retrieved recipes by comparing them to a predefined list of relevant recipes. Precisely, we compute:

- **Hits@k**: measures the percentage of positive examples that appear in the top-k ranked predictions.
- **Mean Reciprocal Rank (MRR)**: represents the average reciprocal rank, calculated by taking the reciprocal of the rank (1/rank) of the first relevant item retrieved.

5.2. Recipe Generation Evaluation

We chose RotatE, the strongest-performing knowledge graph embedding (KGE) model, to bootstrap the recipe generation process. The generation pipeline operates as follows: first, the system retrieves the top-ranking recipes based on aggregated relevance scores. From these, the top 5 most relevant recipes are selected. These retrieved recipes, along with the original user criteria, are then fed into *DeepSeek-R1-0528*, which generates new, tailored recipe suggestions. By leveraging both the user's input and the 5 retrieved examples as context, the model produces a customized recipe.

For comparison, we repeated the process using two alternatives: (1) retrieving the top 5 examples using MiniLM text embeddings, and (2) generating recipes with no example-based retrieval at all.

To evaluate the results, we utilized a separate large language model, *Mistral Small 3.2 24B*, as an LLM-as-a-judge [24, 25] to evaluate the generated recipes according to two criteria: **Satisfiability**

(i.e., whether all specified criteria are met) and **Feasibility** (i.e., whether the recipe can be realistically prepared at home). Each criterion was rated on a scale from 1 (poor) to 5 (excellent), with no additional explanation provided.

We then computed the mean and standard deviation of these evaluation scores and compared the results against baseline RAG methods—specifically, those using only text embeddings and those without either knowledge graph or text embeddings.

6. Results

In this section, we present both the results of our recipe retrieval and generation.

6.1. Knowledge Graph Embedding Evaluation

We conducted a comprehensive evaluation of our knowledge graph embedding models' ability to retrieve relevant recipes that satisfy constraints and compared their performance to two state-of-the-art text embedding models. Specifically, to evaluate retrieval performance, we used the two test datasets described in Subsections 3.2 and 3.3: (1) standard queries without ingredient constraints (criteria-based) and (2) ingredient-only queries (ingredient-based). The results of these experiments are summarized in Tables 4 and 5 by reporting the Hits@k and mean reciprocal rank (MRR) scores.

Table 4
Performance of embedding models on retrieving recipes using Retrieval Test Set 1 (criteria-based).

(a) Results for 1 to 3 user criteria

Model	1 criterion				2 criteria				3 criteria			
	H@1	H@5	H@10	MRR	H@1	H@5	H@10	MRR	H@1	H@5	H@10	MRR
QuatE	97.7	**100**	**100**	98.9	97.0	98.9	98.9	97.9	**91.0**	95.0	95.9	**92.9**
RotatE	**100**	**100**	**100**	**100**	**97.6**	**99.1**	**99.1**	**98.2**	89.2	**95.9**	**96.9**	92.3
TransE	86.4	95.5	97.7	91.4	83.0	93.6	95.2	87.6	66.2	83.5	88.0	74.0
GTE-Large	75.0	90.9	93.2	83.2	59.0	77.7	83.0	67.9	33.3	54.2	62.0	43.3
MiniLM	79.5	93.2	97.7	85.2	34.0	64.4	73.5	47.8	12.7	32.7	43.0	22.4

(b) Results for 4 to 5 user criteria

Model	4 criteria				5 criteria			
	H@1	H@5	H@10	MRR	H@1	H@5	H@10	MRR
QuatE	**75.5**	87.6	90.9	**81.1**	**57.7**	77.7	83.8	**66.8**
RotatE	72.2	**88.5**	**91.8**	79.4	52.9	**77.8**	**83.9**	64.1
TransE	48.1	70.5	77.2	58.3	34.6	58.7	67.5	45.8
GTE-Large	16.4	33.3	41.9	24.9	9.4	21.5	28.5	15.8
MiniLM	4.9	14.0	20.3	0.0	1.8	6.0	9.3	4.5

The results in Table 4 show that knowledge graph embedding models, particularly RotatE and QuatE, significantly outperform both classical (TransE) and language-based baselines (GTE-Large, MiniLM) across all levels of retrieval difficulty. For 1 and 2 criteria, RotatE achieves nearly perfect performance (MRR = 100 and 98.2), indicating that the model has effectively captured semantic structure. QuatE closely follows and even outperforms RotatE on 3-criteria retrieval (MRR = 92.9 vs. 92.3). As the number of criteria increases (4–5), the performance of all models declines, but RotatE and QuatE remain robust, demonstrating strong generalization. In contrast, transformer-based models like MiniLM and GTE-Large show steep performance degradation.

The retrieval performance under ingredient constraints is more challenging across all models compared to Retrieval Test Set 1. Despite this, RotatE consistently outperforms all baselines, achieving nearly perfect results under 1 criterion (MRR = 99.6) and maintaining robust performance even as constraints

Table 5
Performance of embedding models on retrieving recipes using Retrieval Test Set 2 (ingredient-based).

(a) 1 and 2 criteria

Model	1 criterion				2 criteria			
	H@1	H@5	H@10	MRR	H@1	H@5	H@10	MRR
QuatE	88.6	**100**	**100**	93.9	63.9	82.6	87.1	72.6
RotatE	**99.3**	**100**	**100**	**99.6**	**91.5**	**99.4**	**99.8**	**95.1**
TransE	10.0	47.8	63.1	27.2	9.3	25.1	34.5	17.4
GTE-Large	48.1	61.9	68.4	54.8	12.9	28.7	37.1	20.8
MiniLM	30.1	47.8	57.3	38.4	4.6	12.0	16.9	8.7

(b) 3 and 4 criteria

Model	3 criteria				4 criteria			
	H@1	H@5	H@10	MRR	H@1	H@5	H@10	MRR
QuatE	51.5	75.4	83.2	62.4	49.9	76.0	84.3	61.7
RotatE	**79.4**	**96.1**	**98.4**	**86.8**	**72.5**	**93.4**	**96.9**	**81.6**
TransE	7.6	21.8	30.3	15.1	8.7	23.7	32.8	16.7
GTE-Large	7.0	19.1	26.6	13.6	5.5	15.3	22.0	11.1
MiniLM	1.4	4.2	6.4	3.2	0.7	2.1	3.2	1.7

increase. QuatE follows closely, with good results for 1–2 criteria but showing more degradation beyond 3 criteria (MRR = 61.7 with 4 constraints). TransE and the language-based models (GTE-Large, MiniLM) perform significantly worse under increasing constraint complexity.

6.2. Recipe Generation Evaluation

Table 6 presents the mean feasibility and satisfiability scores for recipe generation across three model configurations: RAG with RotatE, RAG with MiniLM, and a baseline without retrieval (No RAG). RAG with RotatE achieves the highest satisfiability score (4.51), indicating that incorporating knowledge graph embeddings contributes positively to producing coherent and satisfying recipes. In contrast, the "No RAG" model slightly outperforms others in feasibility (4.04).

In addition, the table shows the corresponding standard deviations, highlighting that RAG with RotatE also yields the most consistent results, with the lowest variance in both feasibility (0.52) and satisfiability (0.50).

Table 6
Mean and Standard Deviation of feasibility (i.e., whether all specified criteria are met) and satisfiability (i.e., whether the recipe can be realistically prepared at home) scores for recipe generation. Each criterion was rated on a scale from 1 (poor) to 5 (excellent).

Model	**Feasibility (Mean ± SD)**	**Satisfiability (Mean ± SD)**
RAG (RotatE)	4.01 ± 0.52	4.51 ± 0.50
RAG (MiniLM)	3.81 ± 0.94	4.22 ± 0.88
No RAG	4.04 ± 1.07	4.22 ± 1.03

7. Discussion

Retrieval. Across all models, performance declined as the number of criteria increased from one to five, indicating that satisfying multiple user constraints significantly increases the complexity of the retrieval task. Nevertheless, the results underscore the effectiveness of knowledge graph embeddings

(KGEs) in supporting multi-criteria recipe retrieval, while simultaneously revealing the limitations of purely text-based embeddings in such scenarios. Transformer-based models such as MiniLM and GTE-Large exhibited sharp performance drops, suggesting that they struggle to represent structured, combinatorial constraints effectively without domain-specific fine-tuning. These findings highlight the critical role of structured knowledge in enhancing retrieval performance, particularly when multiple constraints must be jointly satisfied.

Generation. In the generation task, the RAG model augmented with RotatE embeddings demonstrated the best balance between quality and consistency. It achieved the highest satisfiability scores while also exhibiting the lowest variance across both feasibility and satisfiability metrics. This suggests that incorporating structured retrieval through KGEs enhances the generation of coherent and dependable recipes. While the "No RAG" baseline showed slightly higher feasibility, it also displayed greater variability, indicating less stable performance. Taken together, these results emphasize the advantage of leveraging structured knowledge representations to improve both the robustness and overall quality of recipe generation under complex conditions.

8. Conclusion

In this work, we introduced RecipeRAG, a novel RAG system that combines knowledge graph embeddings (KGE) with LLMs to enhance recipe generation. We constructed RecipeKG from Food.com and showed that KGE-based retrieval significantly outperforms traditional text-based methods in identifying relevant recipes. Our experiments highlight that RotatE and QuatE embeddings offer superior retrieval performance compared to both classical models like TransE and language-based baselines such as GTE-Large and MiniLM, especially as the number of user constraints increases.

RecipeRAG leverages the structured nature of RecipeKG to retrieve recipes that closely match user preferences and uses LLMs to generate coherent, personalized recipe texts. This integration enables more accurate retrieval and better aligns the generated content with user-defined needs.

Supplemental Material Statement. The dataset, code and relevant material are available at https://github.com/jloe2911/Recipe_RAG.

Declaration on Generative AI

We have used ChatGPT to address the grammatical errors and rephrase the sentences.

References

[1] C. Trattner, D. Elsweiler, Food recommender systems: Important contributions, challenges and future research directions, arXiv preprint arXiv:1711.02760 (2017).

[2] M. Fatemi, Y. Jia, W. Wang, B. Liu, J. Han, Gismo: Graph-based ingredient substitution modeling, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM), ACM, 2022, pp. 2931–2939.

[3] S. Li, Y. Li, J. Ni, J. McAuley, SHARE: a system for hierarchical assistive recipe editing, in: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 2022, pp. 11077–11090. URL: https://aclanthology.org/2022.emnlp-main.761.

[4] W. Min, C. Liu, L. Xu, S. Jiang, Applications of knowledge graphs for food science and industry, Patterns 3 (2022) 100484. doi:10.1016/j.patter.2022.100484.

[5] M. Kazama, M. Sugimoto, C. Hosokawa, K. Matsushima, L. R. Varshney, Y. Ishikawa, A neural network system for transformation of regional cuisine style, Frontiers in ICT 5 (2018) 14. doi:10.3389/fict.2018.00014. arXiv:1705.03487.

[6] S. S. Shirai, O. Seneviratne, M. E. Gordon, C.-H. Chen, D. L. McGuinness, Identifying Ingredient Substitutions Using a Knowledge Graph of Food, Frontiers in Artificial Intelligence 3 (2021) 621766. doi:10.3389/frai.2020.621766.

[7] A. Morales-Garzon, J. Gomez-Romero, M. J. Martin-Bautista, A Word Embedding-Based Method for Unsupervised Adaptation of Cooking Recipes, IEEE Access 9 (2021) 27389–27404. doi:10.1109/ACCESS.2021.3058559.

[8] H. H. Lee, S. Ke, P. Achananuparp, P. K. Prasetyo, Y. Liu, E. Lim, L. R. Varshney, Recipegpt: generative pre-training based cooking recipe generation and evaluation system, Companion Proceedings of the Web Conference 2020 (2020). doi:10.1145/3366424.3383536.

[9] M. Goel, P. Chakraborty, V. Ponnaganti, M. Khan, S. Tatipamala, A. Saini, G. Bagler, Ratatouille: A tool for Novel Recipe Generation, in: 2022 IEEE 38th International Conference on Data Engineering Workshops (ICDEW), 2022, pp. 107–110. doi:10.1109/ICDEW55742.2022.00022. arXiv:2206.08267.

[10] A. Hwang, B. Li, Z. Hou, D. Roth, Large Language Models as Sous Chefs: Revising Recipes with GPT-3, 2023. arXiv:2306.13986.

[11] F. Mohbat, M. J. Zaki, LLaVA-Chef: A Multi-modal Generative Model for Food Recipes, ???? doi:10.1145/3627673.3679562. arXiv:2408.16889.

[12] X. Liu, Y. Feng, J. Tang, C. Hu, D. Zhao, Counterfactual Recipe Generation: Exploring Compositional Generalization in a Realistic Scenario, 2022. arXiv:2210.11431.

[13] B. P. Majumder, S. Li, J. Ni, J. McAuley, Generating personalized recipes from historical user preferences, in: K. Inui, J. Jiang, V. Ng, X. Wan (Eds.), Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 5976–5982. URL: https://aclanthology.org/D19-1613. doi:10.18653/v1/D19-1613.

[14] Y. Liu, Y. Su, E. Shareghi, N. Collier, Plug-and-play recipe generation with content planning, in: A. Bosselut, K. Chandu, K. Dhole, V. Gangal, S. Gehrmann, Y. Jernite, J. Novikova, L. Perez-Beltrachini (Eds.), Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM), Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (Hybrid), 2022, pp. 223–234. URL: https://aclanthology.org/2022.gem-1.19. doi:10.18653/v1/2022.gem-1.19.

[15] A. Pini, J. Hayes, C. Upton, M. Corcoran, AI Inspired Recipes: Designing Computationally Creative Food Combos, in: Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems, ACM, Glasgow Scotland Uk, 2019, pp. 1–6. doi:10.1145/3290607.3312948.

[16] M. Chen, X. Jia, E. Gorbonos, C. T. Hoang, X. Yu, Y. Liu, Eating healthier: Exploring nutrition information for healthier recipe recommendation, Information Processing & Management 57

(2020) 102051. doi:10.1016/j.ipm.2019.05.012.
[17] Y. Pan, Q. Xu, Y. Li, Food Recipe Alternation and Generation with Natural Language Processing Techniques, in: 2020 IEEE 36th International Conference on Data Engineering Workshops (ICDEW), IEEE, Dallas, TX, USA, 2020, pp. 94–97. doi:10.1109/ICDEW49219.2020.000-1.
[18] D. Freedman, P. Diaconis, On the histogram as a density estimator: L2 theory, Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete 57 (1981) 453–476.
[19] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, volume 26, Curran Associates, Inc., 2013.
[20] Z. Sun, Z.-H. Deng, J.-Y. Nie, J. Tang, Rotate: Knowledge graph embedding by relational rotation in complex space, arXiv preprint arXiv:1902.10197 (2019).
[21] S. Zhang, Y. Tay, L. Yao, Q. Liu, Quaternion knowledge graph embeddings, Advances in neural information processing systems 32 (2019).
[22] M. Ali, M. Berrendorf, C. T. Hoyt, L. Vermue, S. Sharifzadeh, V. Tresp, J. Lehmann, PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings, Journal of Machine Learning Research 22 (2021) 1–6. URL: http://jmlr.org/papers/v22/20-825.html.
[23] Z. Li, X. Zhang, Y. Zhang, D. Long, P. Xie, M. Zhang, Towards general text embeddings with multi-stage contrastive learning, 2023. URL: https://arxiv.org/abs/2308.03281. arXiv:2308.03281.
[24] J. Gu, X. Jiang, Z. Shi, H. Tan, X. Zhai, C. Xu, W. Li, Y. Shen, S. Ma, H. Liu, S. Wang, K. Zhang, Y. Wang, W. Gao, L. Ni, J. Guo, A survey on llm-as-a-judge, 2025. URL: https://arxiv.org/abs/2411.15594. arXiv:2411.15594.
[25] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, I. Stoica, Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL: https://arxiv.org/abs/2306.05685. arXiv:2306.05685.