PantryPal: “AI’m Your Chef!”

Nikita Shinde, Weihao Lin, Dylan Lee, Emily Chang
COGS 188: AI Algorithms
UC San Diego

Abstract

PantryPal is an **AI-powered recipe recommendation system** that generates meal ideas based on **user-specified ingredient availability and dietary restrictions**. Traditional recipe recommendation systems often fail to provide **practical meal suggestions**, requiring unavailable ingredients or neglecting user constraints. PantryPal addresses this by integrating **Natural Language Processing (NLP) and filtering** to ensure **feasible, coherent, and adaptable** recipe recommendations.

Our approach includes **ingredient filtering, instruction deduplication, and cooking method classification** using **TF-IDF vectorization and K-Means clustering**. PantryPal successfully refines recipe selection, improves instruction clarity (validated via **BLEU score evaluations**), and organizes recipes based on cooking techniques optimized through learning and validation curves.

PantryPal builds on **AI-driven food recommendation research** but extends usability by explicitly addressing **ingredient constraints and meal preparation challenges**. Future work will focus on **enhancing ingredient recognition, refining cooking steps, and developing a user interface for accessibility**. PantryPal lays the foundation for **personalized and adaptive meal-planning solutions**.

Problem Statement

Traditional recipe recommendation systems often fail to provide practical meal suggestions that align with the user's available ingredients or provide ingredients that the user cannot have. This leads to inefficient meal planning and wasted time.

PantryPal addresses this problem by providing and generating contextually relevant and feasible recipes using Natural Language Processing (NLP) and filtering. The system ensures recipe coherence and adaptability by integrating structured constraints into the generation process.

This problem is quantifiable, as recipe quality can be evaluated using BLEU scores to quantify how well our program preserves the original instructions (text coherence) . It is also measurable, as generated recipes can be compared against reference datasets. Finally, the problem is replicable, as recipe recommendation challenges occur universally and can be applied across different datasets and user needs.

By optimizing meal planning through AI-driven recipe generation, PantryPal enhances accessibility and practicality in everyday cooking.

Introduction

With the increasing demand for personalized meal planning, PantryPal aims to provide an AI-powered recipe generator that suggests meal ideas based on available ingredients, kitchen tools, and time constraints. Traditional recipe recommendation systems often struggle with contextual relevance, offering recipes that require unavailable ingredients or incompatible cooking methods. Our goal is to bridge this gap by integrating Natural Language Processing (NLP) and rule-based filtering techniques to generate recipes that are practical, user-friendly, and adaptable to individual constraints.

Existing research in recipe generation has explored both data-driven and rule-based approaches. Structured datasets such as RecipeNLG have demonstrated the potential of semi-structured text generation (Bień et al., 2020). Deep learning models, such as neural checklist approaches, have shown promise in generating globally coherent recipes by ensuring that generated steps follow a logical sequence (Kiddon et al., 2016). However, these methods often lack adaptability to real-world constraints such as ingredient availability, dietary restrictions, and user preferences. To address these limitations, PantryPal employs a hybrid approach that leverages both machine-learning techniques and rule-based filtering to ensure recipe feasibility and coherence. Prior studies have demonstrated the effectiveness of hybrid systems in food recommendation tasks, showing improvements in adaptability and quality when combining collaborative filtering with rule-based logic (Chow et al., 2023).

PantryPal extracts key cooking actions (verbs) from recipe instructions using NLTK, converts these into numerical features with TF-IDF, and groups recipes with similar methods using K-Means clustering. Rule-based constraints filter out unrealistic ingredient pairings, ensuring that the recommended recipes match user dietary preferences. The system's performance is evaluated using metrics such as BLEU scores to assess the quality of instruction deduplication, and learning/validation curves are employed to analyze the clustering model's behavior. Ultimately, PantryPal seeks to enhance the cooking experience by making meal preparation more accessible, flexible, and personalized..

Data

For this project, we used a dataset from Eight Portions, specifically the *recipes_raw_nosource_epi.json* file. This dataset contains a collection of raw recipe data without source attribution, making it a valuable resource for analyzing and generating recipe recommendations

Dataset Overview:

- **Source**: Eight Portions Recipe Dataset
- **File** Used: *recipes_raw_nosource_epi.json*
- **Size**: The dataset contains approximately 200,000+ recipes.
- **Observations:** Each observation in the dataset represents an individual recipe.
- **Variables:**

- *title*: The name of the recipe.
- *ingredients*: A list of ingredients required for the recipe.
- *instructions*: The steps to prepare the recipe.
- *picture_link* (sometimes missing): URL to an image of the recipe, if available.
    - We are not currently using this but if we were to create a better UI face for PantryPal then we may look into utilizing it.

Data Cleaning and Processing

The raw dataset required several preprocessing steps to make it suitable for use in our system. Below are the key transformations:

1. **Loading the Data:**
    - The JSON file was read into a Pandas DataFrame for easy manipulation.
2. **Handling Missing Values:**
    - Some recipes had missing *title, ingredients,* or *instructions.* Recipes missing critical fields were removed.
3. **Standardizing Ingredients:**
    - Ingredients were converted to lowercase to maintain consistency.
    - Tokenization and lemmatization were applied to normalize ingredient names.
4. **Instruction Cleaning:**
    - Some instructions contained duplicate or redundant steps.
    - A deduplication function was applied to split instructions into sentences and remove repeated steps.
5. **Extracting Cooking Methods:**
    - Natural Language Processing (NLP) techniques were used to extract verbs from instructions, allowing us to categorize recipes based on cooking techniques.
6. **Filtering and Clustering:**
    - TF-IDF (Term Frequency-Inverse Document Frequency) and K-Means clustering were applied to group recipes based on similarities in their cooking methods.

The data cleaning process can be found on our git in our notebooks. The cleaned dataset serves as the foundation for our recommendation system, ensuring that our recipe generation pipeline is both accurate and efficient.

Other Datasets Considered

Initially, we also explored using the following datasets:

- Food.com Recipes and User Interactions (specifically *PP_recipes.csv* and *RAW_recipes.csv*)
- Kaggle Recipe Ingredients Dataset (specifically *test.json* and *train.json*)

However, we encountered challenges in integrating these datasets, such as inconsistencies in formatting, missing values, and difficulties in aligning them with our processing pipeline. Due to these complications, we ultimately decided to work solely with the Eight Portions dataset, as it provided a more structured and reliable foundation for our recipe generation system.

Methods
To evaluate the effectiveness of PantryPal's ingredient filtering, instruction deduplication, and cooking method classification, we implemented a series of experimental tests. Our primary goal was to ensure that the system could accurately filter recipes based on user-defined ingredients, improve instruction readability, and group similar recipes by cooking techniques. We tested our methods using a subset of recipes from the dataset, applying Natural Language Processing (NLP) for text preprocessing, TF-IDF vectorization for feature extraction, and K-Means clustering for organizing recipes. Evaluation metrics such as BLEU scores for instruction clarity and inertia values for clustering performance helped assess the success of our approach. Below, we outline the core methods, evaluation techniques, and results that demonstrate the system's capabilities.

Demo Video of PantryPal
https://www.youtube.com/watch?v=l7fEcr-VJnY&ab_channel=Joon

1. Filtering
1.1 Methods
The method *filter_recipes_by_inclusions_and_exclusions(include, exclude, df)* takes the user's input of ingredients to include and exclude in the recipe and returns the updated data frame with only the filtered recipes. The method iterates through the dataset and keeps recipes if they contain all of the included ingredients and removes them if they contain any of the excluded ingredients. This is the main method that allows us to meet the user's needs regarding ingredient availability as well as potential dietary restrictions.

The method *deduplicate_instructions(instructions)* cleans up the recipe instructions by removing duplicate sentences that may occur due to formatting or data entry errors. Originally, when outputting the instructions for the generated recipes, there were multiple cases of text being duplicated and extra whitespaces being added. Thus, the function was made to clean up these errors. The function splits the instruction text into sentences using regex to detect punctuation followed by whitespace. Afterwards, it iterates over these sentences and strips extra whitespaces, and reassembles the sentences into a string. The function's contribution to the project was to improve the readability and clarity of the generated recipes.

1.2 Evaluation Metrics (how we evaluated if it was working or not)
The *evaluate_deduplication(original, deduped)* method helps us evaluate the effectiveness of the *deduplicate_instructions(instructions)* method by calculating the BLEU scores of both the original and deduplicated instructions. It splits both instructions into sentences using regular expressions that detect punctuation that end a sentence. Then, smoothing is applied to deal with shorter sentences potentially having more overlap, and the BLEU scores are calculated accordingly. This evaluation is an important step to ensure that the redundant instructions are actually removed without affecting the context and main ideas of the instructions. A high score

shows that the deduplication effectively replicated the meaning of the original, while a low score suggests a significant loss or change of meaning.

1.3 Results
For our method *filter_recipes_by_inclusions_and_exclusions(include, exclude, df)*, when tested with sample inclusion criteria (e.g., “chicken”, “carrot”) and exclusion criteria (e.g., “broth”, “onion”), the function successfully returns only recipes that contain all the desired ingredients while filtering out recipes that include unwanted ingredients. In the demo video, the program returned 47 recipes that satisfied the dual requirement.

When generating a recipe given the user constraints the method *deduplicate_instructions(instructions)* successfully outputs the instructions with no extra whitespaces and single instances of each sentence, improving clarity and readability.

2. Cooking Methods
2.1 Methods
The method *extract_verbs(text)* extracts key cooking actions from the instructions by isolating verbs. These verbs provide a summary of the cooking methods used in the recipe. The method uses NLTK’s *word_tokenize* function to break down the instruction text into individual tokens. It then filters out verbs by checking the part of speech (POS) tag assigned to them by nltk.pos_tag. The verbs are then reduced to their base form and concatenated into a single space-separated string. We then utilize these verbs in features like the TF_IDF vectorization to group similar recipes together based on sharing cooking techniques.

TF-IDF Vectorization and K-Means Clustering:
The TfidfVectorizer converts the space-separated verb strings into a matrix that shows the importance of each verb in comparison to the dataset. K-Means is applied to the TF-IDF matrix with a preset of 15 clusters. This algorithm groups recipes such that those within a cluster share similar cooking actions. By clustering recipes, we highlight the patterns in cooking methods across recipes to see which techniques appear together for further improvements and categorization.

The method *generate_recipe(include, exclude, df)* combines the filtering and deduplication functions to generate a complete recipe tailored to the user. First, it applies the filtering function to select recipes matching the user’s inclusion/exclusion criteria. Once the recipes are found it randomly selects one. Then, the instructions are passed through the deduplication function and the final output is formatted to include the recipe name, a list of ingredients, and the cleaned instructions.

The method *plot_learning_curve(df, vectorizer, num_clusters=15, random_state=42)* visualizes how the K-Means clustering performance (measured by inertia) scales with the size of the training data set. The function samples different fractions {10%, 30%, 50%, 70%, and 100%} of the dataset. For each sample, it re-fits the TF-IDF vectorizer and trains a K-Means model. Then, it records the model's inertia (sum of squared distances within clusters). Lastly, it plots the number of training samples against the inertia values. The learning curve helps in understanding whether the clustering model's performance stabilizes as more data is added.

The method *plot_validation_curve(X, cluster_range, random_state=42)* explores the effect of different numbers of clusters on the clustering quality. The method iterates over a range of cluster numbers and for each number it trains a K-Means model on the TF-IDF matrix. Then, it computes and collects the inertia for each model and plots the number of clusters against the inertia. The validation curve assists in hyperparameter tuning for the clustering model. By visualizing how inertia decreases as the number of clusters increases, it helps us identify a balance between model complexity and clustering quality.

2.2 Evaluation Metrics (how we evaluated if it was working or not)

We used the method *evaluate_generation(df, num_samples=10)* to provide an overall measure of how effective our deduplication process is by randomly sampling a number of recipes from a dataset and computing the BLEU score between the original and deduplicated instructions. The function then reports the average BLEU score over the sample. This function provides a metric that is a quantitative measure of how much information is retained after deduplication.

2.3 Results

The TF_IDF transformation successfully converts the extracted verbs from extractor verbs into a matrix and K-Means clusters groups of recipes with similar cooking actions together, as shown in the terminal when running the pantrypal.py file in the video.

The function *generate_recipe(include, exclude, df)* successfully applies the ingredient filtering inputted by the user into the interface and then successfully selects a random recipe from the results. In the video, the program outputted a recipe for Asian Chicken and Cabbage Salad when inputted with the filters to include chicken, carrot and exclude broth, and onion that satisfied all of the constraints given by the user.

As shown in *Figure 1* in the Appendix, The *plot_learning_curve(df, vectorizer, num_clusters, random_state)* outputs a plot that shows that as the number of recipes increases, the total inertia increases roughly linearly.

As shown in *Figure 2*, the *plot_validation_curve(X, cluster_range, random_state)* creates a validation curve to show the inertia decreasing as the number of clusters increases, highlighting that there is a point with an optimal cluster count. The plot supports hyperparameter selection, and helps guide us to the best choice of the number of clusters for the best balance between model complexity and performance.

For our *evaluate_deduplication(original, deduped)* and *evaluate_generation(df, num_samples)* functions, the helper function *evaluate_deduplicataion* computes a BLEU score for each recipe by comparing the original and deduplicated instructions. *evaluate_generation* then aggregates the scores over several samples, providing the average BLEU score. In the video, the program outputted that over 10 random recipes, the average BLEU score was found to be 0.3743, indicating that the deduplication process is removing a significant amount of redundancy.

Discussion

Interpreting the Results

Our integrated program, PantryPal, effectively combines ingredient filtering, instruction deduplication, verb extraction, and clustering to generate clear, tailored recipes for the user. The results demonstrate that each component works together to ensure that only recipes meeting the user-specified criteria are selected, that instructions are clean and concise, and that recipes are organized by cooking techniques. This approach meets the practical needs of the user but also provides insight into the underlying structure of the recipe dataset.
The filtering function reliably returns recipes that contain all desired ingredients while excluding those with unwanted items. Our tests shown in the video demonstrate that when users specify inclusion and exclusion criteria, only recipes meeting these criteria are selected, meeting the needs of dietary restrictions or available ingredients. Achieving these standards is essential for providing personalized recipe recommendations.
The deduplication method successfully removes redundant sentences from recipe instructions, resulting in clear and concise outputs. The evaluation metric of BLEU scores in the results confirms that the deduplication process retains key content while sufficiently eliminating repetition. This improvement in instruction clarity makes the recipes overall easier to follow and enhances user satisfaction.
By extracting cooking verbs and applying TF-IDF and K-Means clustering, our program groups recipes with similar cooking methods. The clusters, characterized by their top representative verbs, organize recipes by common culinary techniques. Additionally, the learning and validation curves, in *Figure 1* and *Figure 2*, support model selection by showing how inertia scales with training sample size and how the number of clusters affects performance, leading us to an optimal model configuration.

Limitations

While PantryPal successfully generates recipes based on available ingredients, several limitations affected its implementation and current functionality.

1. Data Availability and Quality
Our system relies on *recipes_raw_nosource_epi.json*, which contains a large number of recipes but is not perfectly structured. Some entries have missing instructions, inconsistent ingredient formatting, and unstructured text, which require additional processing. While preprocessing techniques such as deduplication and NLP-based ingredient standardization helped, the lack of standardized ingredient representations limited the accuracy of filtering and clustering. Expanding to more structured datasets, user-submitted recipes, or social media food posts could improve robustness, but this would introduce additional challenges in data cleaning, normalization, and bias mitigation.
2. Computational Constraints and Hyperparameter Optimization
Due to time and computational limitations, we optimized only a limited set of hyperparameters for training and fine-tuning. While the TF-IDF vectorization and K-Means clustering approach was effective, a more extensive hyperparameter search exploring alternative learning rates, tokenization strategies, and decoding techniques (e.g., beam search or nucleus sampling) could improve the fluency, coherence, and practicality of generated recipes. Additionally, optimizing the trade-off between response speed and recipe complexity remains an area for future exploration.
3. Ingredient Filtering Constraints
The current implementation allows users to specify ingredients to include and exclude, but filtering is based on exact keyword matching. This means that slight variations in ingredient naming (e.g., "bell pepper" vs. "capsicum") may lead to unintended exclusions. Furthermore, our system does not provide ingredient substitutions when an excluded ingredient is a key component of a recipe, potentially limiting the number of viable recommendations. A more advanced semantic matching model could improve ingredient recognition and flexibility.
4. Rule-Based Filtering Limitations
While rule-based filtering helps ensure feasibility, it also imposes rigid constraints that may over-restrict creativity and adaptability. Fixed rule sets may fail to accommodate:

- Unconventional ingredient pairings (e.g., fusion cuisine)
- Regional cooking styles (e.g., culturally specific ingredient combinations)
- Evolving food trends (e.g., plant-based alternatives, viral recipes) A more dynamic rule-based system incorporating reinforcement learning or personalized preference modeling could allow for greater adaptability while still maintaining practicality.

5. Recipe Instruction Processing Limitations
We implemented a deduplication method to remove redundant instructions, but the system does not restructure or refine step-by-step directions. Some recipes still contain awkward phrasing or fragmented steps, as instructions in the dataset vary significantly in format and detail. Implementing sequence modeling techniques or refining instructions using natural language generation (NLG) methods could improve coherence and readability.
6. Cooking Method Classification Challenges

We extracted cooking verbs from recipe instructions to cluster recipes based on cooking techniques, but the system lacks contextual awareness. For example, terms like "bake" and "roast" might be grouped despite being distinct processes. Additionally, recipes often use ambiguous or informal phrasing, making it difficult to consistently categorize cooking methods. A more structured ontology of cooking techniques or context-aware NLP models could improve classification accuracy.

7. No Personalization or Feedback Mechanism

PantryPal cannot currently adapt to user preferences over time. Users cannot rate generated recipes or adjust recommendations based on past interactions. While we successfully implemented ingredient-based filtering, incorporating a feedback-based preference learning system would enable PantryPal to provide more personalized recommendations based on historical selections.

8. Lack of a User Interface

Currently, PantryPal operates as a command-line interface (CLI), which limits accessibility for non-technical users. While the functionality is effective, a graphical user interface (GUI) or web-based application would make the system more intuitive, visually appealing, and interactive, allowing users to input preferences more easily and explore recipe recommendations in a structured format.

Addressing these limitations would refine PantryPal’s effectiveness, improving both the quality of generated recipes and the overall user experience while making the system more flexible, scalable, and adaptable to real-world cooking scenarios.

Ethics and Privacy

PantryPal, like any AI-driven recommendation system, raises ethical and privacy concerns related to bias, security, user safety, and misinformation. Addressing these challenges is essential to ensure fair, private, and responsible Ai-driven recipe recommendations.

1. Bias and Inclusivity

Our model is trained on existing recipe datasets, which may inherit biases present in the data, including:

- **Cultural bias in ingredient pairings** – The dataset may overrepresent certain cuisines while underrepresenting others, leading to a lack of inclusivity in recipe recommendations.
- **Dietary accessibility limitations** – Common dietary preferences such as vegan, halal, kosher, or allergen-free meals may not be adequately represented, reducing the system’s ability to accommodate diverse users.
- **Economic constraints** – Many recipes assume access to expensive or specialty ingredients, making recommendations less practical for users with budgetary restrictions.

To mitigate bias, Pantry Pal should incorporate diverse datasets that represent a broad range of cuisines, dietary needs, and economic accessibility. Future iterations could allow users to explicitly select cultural preferences, dietary restrictions, and budget constraints to ensure the recommendations align with their individual needs.

2. Privacy and Data Security
Pantry Pal requires user inputs such as ingredient lists, dietary preferences, and kitchen tools, which could reveal sensitive information related to religious beliefs, allergies, or medical conditions. Unauthorized access to this data could pose privacy risks.

To enhance privacy and security, Pantry Pal should adopt the following measures:

- **Minimal Data Retention** – User inputs should be processed locally or securely stored with encryption to minimize exposure.
- **No Persistent User Profiles** – Unless explicitly opted in, user data should not be stored or linked to identities.
- **Anonymization Techniques** – Any stored data should be anonymized to prevent the identification of individual users.
- **Secure Data Handling** – Adopting best practices for encryption, access controls, and secure transmission protocols can help safeguard user data from breaches.

3. Health and Nutritional Safety
AI-generated recipes might suggest meals that are nutritionally imbalanced or fail to accommodate critical dietary restrictions, leading to potential health risks. Users with food allergies, chronic illnesses, or specific nutritional goals could receive inappropriate or even harmful meal suggestions.

To address this, Pantry Pal should integrate:

- **Explicit Dietary Filters** – Users should be able to specify dietary restrictions (e.g., gluten-free, low-sodium, diabetic-friendly) to ensure all recommendations are safe.
- **Nutrition-Aware Filtering** – Recipes should be evaluated against basic nutritional guidelines to avoid suggesting meals that are excessively high in unhealthy fats, sugars, or sodium.
- **Allergen Detection** – Clearly identifying potential allergens in recipes can prevent unintended exposure to harmful ingredients.

4. Safety of Cooking Instructions
AI-generated recipes could produce unclear, misleading, or unsafe cooking instructions due to inconsistencies in the dataset. Risks include:

- **Unclear cooking steps** – Vague or incomplete instructions may confuse users, leading to mistakes in preparation.

- **Unsafe food handling** – Missing or incorrect safety instructions (e.g., failing to specify minimum cooking temperatures for meat) could result in foodborne illness.
- **Misinformation risks** – Some generated recipes might include impractical or physically impossible steps due to errors in text generation.

To mitigate these risks, Pantry Pal should implement:

- **Instruction Validation** – Rule-based systems can check for missing or unsafe instructions (e.g., ensuring meat is always cooked to a safe temperature).
- **Structured Step Formatting** – Clearly breaking down instructions into well-defined steps can improve user comprehension.
- **Warnings for Risky Instructions** – If an instruction appears ambiguous, Pantry Pal should flag it and provide safer alternatives.

To ensure Pantry Pal is ethical, private, and safe, future iterations should focus on bias reduction, enhanced data security, health-conscious filtering, and validated cooking instructions. These improvements will make the system more inclusive, responsible, and trustworthy for all users.

Future Work

While PantryPal Successfully displays and generates recipes based on the user's inputs of included and excluded ingredients, there are several areas for improvement and future exploration. Here we will go further, than we did in the previous sections, into the details of what could be done to improve PantryPal.

1. Improving Ingredient Understanding: Matching and Substitutions

Currently, ingredient filtering is based on exact text matches, which can lead to missed results if users input synonyms or variations (e.g., "bell pepper" vs. "capsicum"). Future work could integrate word embeddings or semantic similarity models to improve ingredient recognition and enable intelligent ingredient substitutions when a required ingredient is unavailable.

2. Enhancing Recipe Instruction Coherence

Our deduplication process removes redundant steps from instructions, but it does not refine step transitions or ensure logical ordering. Future improvements could involve sequence modeling or attention-based models to structure instructions more naturally and enhance readability.

3. More Advanced Filtering for Excluded Ingredients

Currently, excluded ingredients completely remove recipes from consideration, which may reduce the number of viable options. Future iterations could incorporate ingredient replacement mechanisms, suggesting alternative ingredients instead of outright filtering recipes.

4. Expanding Cooking Method Understanding
We extract cooking verbs using NLP to identify different cooking methods, but the system currently treats these verbs as isolated terms. A more structured hierarchical clustering approach could group recipes by cooking technique, providing better categorization and recommendations based on user preferences.

5. Incorporating Nutritional and Dietary Constraints
Right now the model does allow for the user to exclude certain ingredients. However, currently, the dataset lacks structured nutritional information, making it difficult to filter recipes by dietary needs. Future improvements could involve integrating external nutritional databases or applying predictive modeling to estimate nutritional values for recipes.

6. Refining Clustering and Recipe Categorization
Our K-Means clustering approach groups recipes based on cooking method similarities. However, it does not currently consider ingredient composition. Future versions could explore hierarchical clustering or multi-factor categorization, incorporating both cooking methods and ingredient profiles to improve recipe classification.

7. Allowing Users to Specify Additional Constraints
Currently, users can filter recipes based on ingredient inclusion and exclusion, but additional inputs such as maximum cooking time, preferred cooking method (e.g., baking, stovetop, grilling), or dietary preferences could enhance the recommendation process. Future iterations could incorporate these parameters to provide more tailored recipe suggestions.

8. Developing a User Interface for PantryPal
Currently, PantryPal operates via a command-line interface, which may not be accessible to all users. Future work could involve designing a graphical user interface (GUI) or web-based application, allowing users to input their ingredient preferences, filtering options, and receive visually appealing, interactive recipe recommendations.

By addressing these limitations, PantryPal can evolve into a more comprehensive and intelligent recipe recommendation system, offering users more accurate, adaptable, and personalized meal suggestions.

Final Results
PantryPal successfully integrates NLP and rule-based filtering to generate recipes tailored to user-defined ingredient availability and dietary restrictions. Our results demonstrate that the hybrid approach effectively improves the practicality, coherence, and adaptability of recipe recommendations and addresses the limitations of traditional recipe recommendation systems. Our key findings include:

1. Effective Ingredient Filtering and Recipe Selection
The *filter_recipes_by_inclusions_and_exclusions* function accurately selects recipes that match user-specified ingredient requirements, ensuring meal feasibility while accommodating dietary restrictions. In testing, the system successfully retrieved 47 recipes that met the inclusion/exclusion criteria (e.g., "chicken," "carrot") and excluded unwanted ingredients (e.g., "broth," "onion") which demonstrates the ability to refine recommendations based on user constraints. While effective, this approach currently relies on exact text matching, meaning that synonyms (e.g., "bell pepper" vs. "capsicum") may lead to unintended exclusions. Future implementations could improve flexibility through semantic similarity models (e.g., word embeddings).
2. Improved Recipe Instruction Coherence
The *deduplicate_instructions* function effectively removed redundant steps, enhancing clarity and readability, as we saw that many recipes in the dataset contained repetitive, redundant, or inconsistent steps, making them harder to follow. Evaluations using BLEU scores showed that the deduplication process preserved the meaning of instructions while reducing unnecessary repetition, achieving an average BLEU score of 0.3743 across sampled recipes.This function significantly improves the readability and usability of AI-generated recipes, ensuring that cooking steps remain clear and easy to follow.
3. Cooking Method Categorization Using NLP & Clustering:
Extracting cooking method verbs and applying TF-IDF vectorization with K-Means clustering enabled us to categorize recipes based on shared cooking techniques. The *extract_verbs* function was used to identify cooking actions from recipe instructions, extracting verbs such as “stir,” “boil,” “fry,” and “bake.” Using TF-IDF vectorization, each recipe was numerically represented based on the importance of its cooking actions, allowing for meaningful comparisons. K-Means clustering was then applied, grouping recipes into 15 clusters based on similar cooking methods. Recipes grouped within the same clusters shared similar cooking techniques, demonstrating that the model successfully identified cooking method similarities. The learning curve analysis revealed that as dataset size increased, the clustering model's performance stabilized, validating that a larger dataset would further improve accuracy. The validation curve showed that inertia decreased as the number of clusters increased, confirming that 15 clusters was a reasonable choice for balancing granularity and performance.
4. Hyperparameter & Model Performance Insights:
Due to computational constraints, only a limited set of hyperparameters were explored. However, we tested different variations of TF-IDF vectorization parameters (e.g., stopword removal, n-gram range), number of clusters in K-Means (tested values: 10, 15, 20), and BLEU score evaluation for instruction deduplication. Our hyperparameter tuning efforts for TF-IDF and K-Means clustering revealed that increasing dataset size improves classification but requires more computational resources. The validation curve confirmed that inertia (sum of squared distances within clusters) decreases as the number of clusters increases, highlighting an optimal point for balancing clustering granularity and efficiency.

5. Overall Effectiveness of PantryPal's Recipe Generation Pipeline

Ingredient filtering successfully refines recipe selections, ensuring that only feasible meal options are presented to users. Instruction deduplication improves recipe readability and coherence, as confirmed by BLEU score evaluations. Cooking method clustering effectively categorizes recipes based on similar techniques, enhancing meal organization and usability.

PantryPal's approach aligns with existing research on AI-driven food recommendation systems, such as RecipeNLG (Bień et al., 2020) and hybrid filtering models (Chow et al., 2023), which demonstrate the effectiveness of combining structured constraints with machine learning to enhance adaptability. However, our work extends beyond prior studies by explicitly addressing real-world constraints, usability, and filtering mechanisms to improve practical meal planning.

Moving forward, enhancing ingredient recognition, refining cooking step structuring, and incorporating user feedback loops will further improve personalization and adaptability. Future work may explore real-time nutritional analysis, reinforcement learning for dynamic recipe refinement, and an interactive user interface (UI) to improve accessibility. PantryPal lays the groundwork for more intelligent, accessible, and personalized meal-planning solutions.

Citations

Bień, Michał, et al. “RecipeNLG: A Cooking Recipes Dataset for Semi-Structured Text Generation.” Proceedings of the 13th International Conference on Natural Language Generation, 2020, aclanthology.org/2020.inlg-1.4/, https://doi.org/10.18653/v1/2020.inlg-1.4.

Kiddon, Chloé, et al. Globally Coherent Text Generation with Neural Checklist Models. Association for Computational Linguistics, 2016.

Teng, Chun-Yuen, et al. “Recipe Recommendation Using Ingredient Networks.” Proceedings of the 4th Annual ACM Web Science Conference, 22 June 2012, pp. 298–307, www.researchgate.net/publication/51958810_Recipe_recommendation_using_ingredient_networks, https://doi.org/10.1145/2380718.2380757.

Chow, Y.-Y., et al. “Food Recommender System: A Review on Techniques, Datasets and Evaluation Metrics.” Journal of System and Management Sciences, vol. 13, no. 5, 28 Sept. 2023, www.semanticscholar.org/paper/Food-Recommender-System%3A-A-Review-on-Techniques%2C-Chow-Haw/c51a3ad0e484a19687f2415cd4e7786b9c7c6d00, https://doi.org/10.33168/jsms.2023.0510. Accessed 19 Mar. 2025.

Appendix

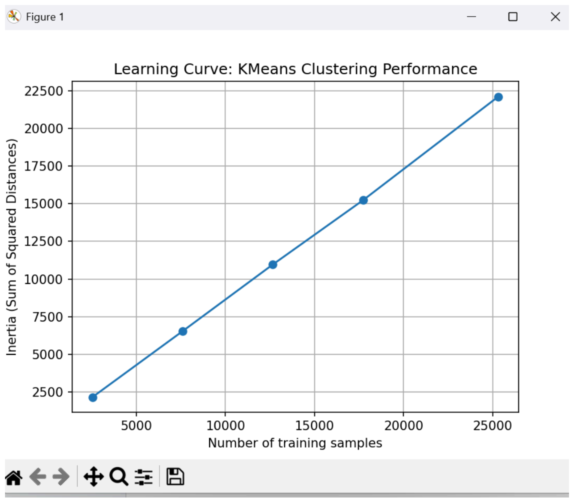

Figure 1. Learning Curve KMeans Clustering Performance

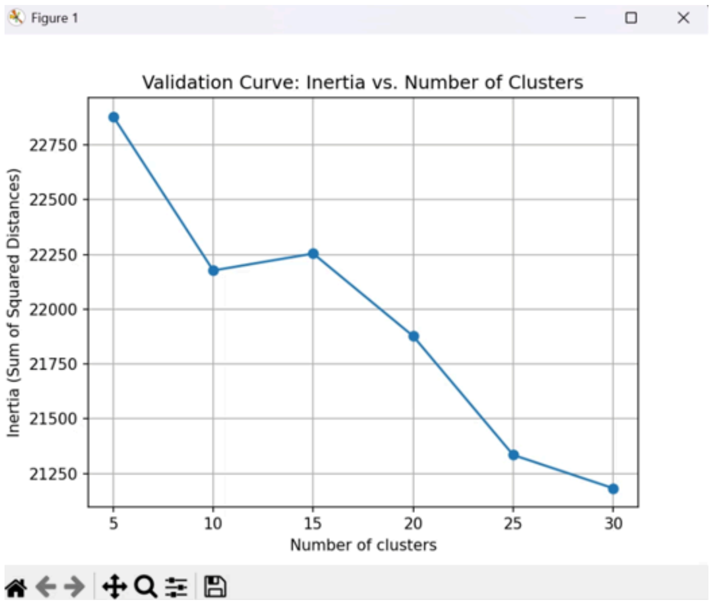

Figure 2: Validation Curve Inertia vs. Number of Clusters