Final Project Proposal: Recipe Retrieval System Based on Available Ingredients

Edward Lu, Zihan Liu, Xulun Luo

1. Introduction and Problem Statement

Every day, households worldwide discard unused ingredients while struggling to find creative ways to prepare meals. The challenge of matching available ingredients to recipes is universal, yet existing tools often fail to account for partial matches, ingredient substitutions, or semantic variations in cooking terminology. This gap limits users' ability to reduce food waste and explore diverse culinary options.

Our project addresses the problem of inefficient recipe discovery by designing a system that retrieves relevant recipes based on a user's available ingredients. The core challenge lies in interpreting incomplete or variable ingredient lists (e.g., "tomatoes" vs. "tomato paste") and ranking recipes by practicality and appeal. Successfully solving this problem would empower users to cook more sustainably and creatively, directly contributing to global efforts to reduce food waste.

2. Team Members & Roles

Zihan Liu: Programming/Software Development
Responsibilities: System implementation, algorithm development, and data pipeline construction
Xulun Luo: Evaluation
Responsibilities: Designing test metrics, conducting error analysis, and ensuring scientific validation
Edward Lu: Software Development & Technical Writing
Responsibilities: Algorithm development, documentation, paper writing, and project management
Our Collaboration: We will hold Weekly synchronizations with our mentor. Also, we will use GitHub for version control.

3. Related Works

We analyze prior work in recipe retrieval and ingredient-based systems, focusing on techniques, evaluation practices, and dataset usage relevant to our project. The followings are the related works:

Recipe Recommendation System Using TF-IDF: This paper "Recipe Recommendation System Using TF-IDF" proposes a mobile application that allows users to search for recipes based on ingredients they have available. The system employs content-based recommendation using Term Frequency-Inverse Document Frequency (TF-IDF) and Cosine Similarity to analyze a dataset of Indian cuisine recipes. Users can filter recipes by course type and diet type, providing a personalized cooking experience based on ingredient availability.

SHARE: a System for Hierarchical Assistive Recipe Editing: This paper presents SHARE, a system designed for hierarchical ingredient-to-step generation, explicitly decoupling the base recipe from edited instructions. This approach allows for the creation of more diverse and personalized recipes without requiring paired data, leveraging a large corpus of recipes for training. SHARE's hierarchical approach to recipe editing can inform our system's design, particularly in managing the relationship between ingredients and preparation steps when adapting recipes based on available ingredients.

Semantic-aware Transformation of Short Texts Using Word Embeddings: This study explores methods for adapting recipes by replacing or adding ingredients based on specific criteria, such as dietary preferences. The approach utilizes word embeddings to understand semantic relationships between ingredients, facilitating meaningful substitutions. Incorporating semantic understanding of ingredient relationships can enhance our system's capability to suggest viable substitutions, accommodating users' available ingredients and dietary restrictions.

Improved Instruction Ordering in Recipe-Grounded Conversation: The paper investigates recipe-grounded conversational agents, focusing on improving the ability to provide cooking instructions in the correct order. Through analysis of GPT-J's output, the authors identified that the primary challenge in recipe-grounded dialogue systems is maintaining proper instruction sequencing. Their manual evaluation showed that among incorrect responses, wrong instruction ordering was the most common error type.

Generating Informative Conversational Response using Recurrent Knowledge-Interaction and Knowledge-Copy: This paper explores knowledge-driven dialogue systems, focusing on the challenge of integrating multiple knowledge sources into conversational responses while maintaining coherence. The authors identify two key limitations in existing approaches: pipeline methods that separate knowledge selection from generation, and joint methods that use static knowledge information which loses confidence during decoding. Their solution, KIC (Knowledge-Interaction and Knowledge-Copy), introduces a recurrent knowledge interaction mechanism that dynamically updates knowledge attention during response generation, paired with a knowledge-aware pointer network for copying terms directly from knowledge sources.

4. Methodology

4.1 Data Collection and Cleaning

Here is the dataset: RecipeNLG (recipes with structured ingredients and steps).
Here are the cleaning steps:

1. Normalization: Standardize ingredient names (e.g., "tomato paste" → "tomato") using rules from "Rule-Based Ingredient Substitution".
2. Annotation:
    - Label "core" vs. "optional" ingredients (e.g., "chicken" is core for "Chicken Curry"; "cilantro" is optional).
    - Tag dietary preferences (vegetarian, gluten-free) and course types (main, dessert).
3. Splitting: Train (70%), Development (15%), Test (15%) sets, with strict separation.

4.2 Baseline Model (TF-IDF) and Enhancement

Our baseline system builds on the TF-IDF framework introduced in class. We will implement a standard TF-IDF vectorizer that treats each recipe's ingredient list as a document and individual ingredients as terms. The system computes cosine similarity between the user's ingredient query vector and all recipe vectors in our database. Basic preprocessing includes text normalization (standardizing ingredient names like "tomato paste" → "tomato") and removal of measurements (e.g., "1 cup"). This creates an efficient first-pass retrieval system capable of handling large recipe databases.

For enhancement, we will implement a multi-layer refinement system that extends beyond simple term matching:

1. **Query Expansion**: The system identifies ingredient synonyms and functional substitutes through a culinary knowledge base (e.g., mapping "dairy milk" to alternatives like "almond milk" or "soy milk").

2. **Hierarchical Processing**:

    - First pass: Identify broadly matching recipes through standard TF-IDF
    - Second pass: Apply finer-grained filters based on ingredient coverage percentages and compatibility scores

3. **Intelligent Substitution**: Maintain a substitution dictionary that preserves recipe integrity when suggesting adaptations, ensuring recommendations remain within culinary boundaries.

4. **Weighted Ingredient Importance**: Distinguish between "core" and "optional" ingredients, giving higher weight to matches on core ingredients.

5. Evaluation

Our evaluation framework employs a multi-stage assessment process measuring both technical performance and practical utility for recipe recommendation based on available ingredients.

5.1 System Outputs

The system generates three key outputs:

1. A ranked list of top 5 recipe recommendations
2. Suggested ingredient substitutions when exact matches aren't available
3. An ingredient coverage percentage for each recommendation

5.2 Evaluation Metrics

We will employ three principal metrics:

1. **Precision@5**: Our primary measure, calculating the percentage of truly relevant recipes in the top 5 recommendations. A recipe is considered relevant if it has either:
   - ≥80% exact ingredient matches, or
   - ≥90% valid substitutions for core ingredients
2. **Ingredient Utilization Score (IUS)**: A weighted assessment of how effectively the system uses available ingredients:
   - Core ingredients: 70% weight
   - Optional ingredients: 30% weight
3. Formula: IUS = (0.7 × % core ingredients used) + (0.3 × % optional ingredients used)

4. **Substitution Validity Rate (SVR)**: Evaluates the culinary appropriateness of suggested ingredient replacements through expert review on a scale of 1-5, where 5 represents a perfect substitution.

5.3 Dataset Partitioning and Cross-Validation

We maintain strict data partitioning using the Recipe1M+ dataset:

- Training set: 700,000 recipes (70%) - For building TF-IDF vectors and substitution models
- Development set: 150,000 recipes (15%) - For parameter tuning and error analysis
- Test set: 150,000 recipes (15%) - Exclusively for final evaluation

During development, we will implement 5-fold cross-validation to optimize system parameters including:

- n-gram ranges

- Stop words
- Substitution thresholds
- Coverage weightings

5.4 Error Analysis

We will conduct structured error analysis on 50 development set queries sampled across cuisine categories. Each error case will undergo thorough investigation to identify root causes, including:

- Gaps in the ingredient knowledge graph
- Algorithm limitations
- Threshold misconfigurations

Findings from this analysis will directly inform iterative system improvements, with changes validated against fresh development samples before final testing.

6. Task Division and Timeline

Our three-member team will execute parallel development tracks over a one-month period:

6.1 Team Roles and Responsibilities

- **Zihan Liu**: Lead system implementation and algorithm development
  - TF-IDF core functionality
  - Semantic enhancements and query expansion
  - Integration of components
- **Xulun Luo**: Evaluation and testing lead
  - Test metric implementation
  - Comprehensive error analysis
  - Test case preparation and validation
- **Edward Lu**: Algorithm development and technical documentation
  - Substitution dictionary development
  - Technical documentation
  - Project management and coordination

6.2 Week-by-Week Timeline

Week 1: Setup and Planning

- Joint finalization of dataset preparation and cleaning (All)
- Establish evaluation framework and metrics (Xulun, Edward)
- Set up GitHub repository and development environment (Zihan)
- Develop initial data preprocessing pipeline (Zihan, Edward)

Week 2-3: Core Development

- Build basic TF-IDF retrieval system (Zihan)
- Develop ingredient substitution dictionary (Edward)
- Prepare test cases and evaluation scripts (Xulun)

- Implement query expansion module (Zihan, Edward)
- Document implementation details and design decisions (Edward)

Week 3-4: Enhancement and Integration

- Integrate query expansion with base retrieval system (Zihan)
- Implement weighted ingredient importance (Edward)
- Conduct preliminary evaluation on development set (Xulun)
- Begin error analysis and system optimization (All)

Week 4: Finalization

- Complete system optimization based on error analysis (Zihan, Edward)
- Conduct final evaluation on test set (Xulun)
- Prepare final documentation and presentation (All)
- Code cleanup and final GitHub submission (Zihan)

6.3 Coordination Mechanisms

- Weekly synchronization meetings with project mentor
- Continuous integration through GitHub pull requests and code reviews
- Shared documentation on Google Docs for real-time collaboration
- Clear interfaces between components to enable parallel development
- Buffer days (3-5) at the end for troubleshooting and integration issues

This structure maintains individual ownership while enabling collaborative problem-solving through shared code reviews and regular standups. Each component will have clearly defined input/output specifications to minimize dependencies between team members' work.