Recipe Retriever: A Weighted TF–IDF + Sentence–BERT Hybrid for Ingredient–Aware Search

Xulun Luo
New York University
x13874@nyu.edu

Zihan Liu
New York University
zl4701@nyu.edu

Edward Lu
New York University
gl2576@nyu.edu

Abstract

Home cooks often struggle to decide what to cook with ingredients already on hand, leading to an estimated ~150 kg of edible food waste per U.S. household annually. We introduce *ingredient–aware recipe retrieval*: given a list of available items, return a recipe that requires no additional ingredients. Our system **Recipe Retriever** first ranks candidates using a weighted TF–IDF model and then re-ranks the top results with Sentence–BERT cosine similarity and an ingredient-coverage filter. Evaluated on a cleaned 2.2 M-recipe subset of RECIPENLG, the final model attains **91.1% accuracy** on 45 held-out queries—improving the baseline TF–IDF by 20 percentage points—while maintaining a median CPU latency of 35 ms per query. We release code, data splits, and evaluation scripts to foster sustainable cooking research.

1 Introduction

Food waste contributes significantly to global greenhouse-gas emissions, with many perishables discarded because home cooks face decision fatigue when trying to plan meals around existing ingredients. Choosing what to cook becomes a daily challenge that often leads to unnecessary food waste and additional grocery store trips.

We address *ingredient-aware retrieval*: retrieving a single recipe whose ingredient list is fully covered by user-supplied items (allowing universal staples like salt). This poses three challenges:

C1 Sparse vs. dense signals. How to combine lexical precision with semantic synonymy?

C2 Section importance. Titles often carry key terms; directions are verbose. How should weights be distributed?

C3 Efficiency. The system should run on CPUs in smart fridges or mobile devices.

Contributions.

- A thoroughly cleaned, diet-tagged edition of RECIPENLG (2.20 M recipes).
- A weighted TF-IDF scheme with default weights of 2.0/1.5/0.5 for title/ingredients/directions.
- A hybrid sparse+dense pipeline that achieves 0.911 accuracy (+20 pp from 0.711 baseline).
- Progressive system evolution across versions V1-V4 with substantial accuracy improvements.

The rest of the paper is structured as follows: Section 2 surveys related work; Section 3 explains the background techniques; Section 4 presents the corpus; Section 5 describes the retrieval pipeline; Section 6 details experimental setup; Section 7 reports results; Section 8 analyses errors and limitations; Section 9 concludes and Section 10 proposes future directions.

2 Related Work

2.1 Sparse Retrieval

The vector-space model with TF–IDF weighting remains a strong baseline for information retrieval. BM25 (Best Matching 25) is an improved ranking function that extends the basic TF-IDF model by adding term frequency normalization and document length normalization factors. It has proven effective in many retrieval tasks. More recently, TF–IDF has been applied to Indian-cuisine recommendation, but such approaches typically lack synonym handling or hybrid ranking techniques.

2.2 Dense Representations

Word embeddings such as Word2Vec and transformer-based models like BERT capture semantic relations between terms that sparse models cannot. Recent work like SHARE uses hierarchical encoders for recipe editing, showcasing the potential of dense representations for modeling complex food relationships. Other research has exploited embeddings for allergen substitutions, demonstrating their effectiveness for ingredient relationships. We adopt Sentence–BERT for its optimal speed/quality trade-off in the recipe domain.

2.3 Ingredient Networks

Ingredient relation graphs reveal both complementary and substitutable flavor pairings based on chemical composition analysis. We borrow synonym lists and relationship mapping approaches from this research to normalize regional ingredient names and build our substitution graph.

3 Background

3.1 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF is a fundamental information retrieval technique that quantifies the importance of words in a document collection. It consists of two key components:

- **Term Frequency (TF)**: Measures how frequently a term appears in a document, calculated as:

$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{1}$$

  where $f_{t,d}$ is the count of term t in document d.

- **Inverse Document Frequency (IDF)**: Penalizes terms that appear in many documents, calculated as:

$$\mathrm{idf}(t,D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \tag{2}$$

  where $|D|$ is the total number of documents and $|\{d \in D : t \in d\}|$ is the number of documents containing term t.

The final TF-IDF score is the product of these components. In recipe recommendation systems, TF-IDF has been successfully applied to ingredient matching. For example, Chhipa et al. implemented a recipe recommendation system using TF-IDF and cosine similarity for Indian cuisine. Their approach enabled users to search for recipes based on available ingredients and filter results by course type and diet requirements. TF-IDF excels at exact matching but struggles with semantic relationships between ingredients, necessitating complementary approaches for robust retrieval.

3.2 BERT and Sentence-BERT

BERT (Bidirectional Encoder Representations from Transformers) revolutionized natural language processing by introducing deeply contextualized word representations. Unlike previous word embedding methods, BERT captures the context-dependent meaning of words through a transformer architecture trained on vast text corpora via masked language modeling.

Sentence-BERT (SBERT) adapts the original BERT model specifically for sentence similarity tasks. It employs a siamese network structure where sentence pairs are encoded independently, producing fixed-size embeddings that can be compared using cosine similarity. This modification enables much faster similarity computation (reducing time complexity from quadratic to linear) while maintaining high-quality semantic representations.

In our recipe retrieval system, Sentence-BERT provides the semantic understanding needed to recognize that ingredients like "scallion" and "green onion" are effectively identical, despite having different lexical forms. This capability complements TF-IDF's precision with crucial flexibility for ingredient matching.

4 Data

4.1 Corpus Construction

The RECIPENLG dataset used in our research comprises over 2.2 million cooking recipes. We performed a methodical cleaning process to ensure data quality:

1. **Field sanity checks:** We removed recipes missing any core field (title, ingredients list, or directions).

2. **NER merging:** Using spaCy's named entity recognition, we joined partial ingredient spans (e.g., "fresh" + "tomatoes" $\rightarrow$ "fresh tomatoes") to create more coherent ingredient entries.

3. **Diet tagging:** We implemented rule-based inference for vegan, keto, and gluten-free classifications based on ingredient lists, enabling dietary filtering.

As shown in Table 1, the corpus contains 2.20 million recipes spanning diverse cuisines, dietary preferences, and complexity levels.

Processing Step	Recipes	Avg. Tokens
Raw dataset	2.20 M	128
+ sanity filter	2.20 M	121
+ NER merge	2.20 M	119
+ diet tags	2.20 M	119

Table 1: Corpus statistics after each cleaning stage.

4.2 Evaluation Queries

To assess system performance, we curated a set of 45 realistic queries containing 3–6 ingredients each. These queries were created by the authors and categorized into three groups: protein-centered (15), vegetable-centered (15), and pantry staples (15). All evaluation queries were withheld from model training to ensure unbiased performance assessment. Hyperparameter tuning was performed using a small development subset of 15 queries (5 from each category), which was kept separate from our final 45-query test set.

5 Methodology

Our Recipe Retriever system employs a multi-stage pipeline combining traditional sparse retrieval with neural semantic matching, as illustrated in Figure 1.

5.1 Weighted Sparse Ranking

We tokenize each recipe into three distinct fields: title (T), ingredients (I), and directions (S). Recognizing the different importance of these sections, we apply a weighted TF–IDF scoring approach. With TF–IDF weight $w(t, d)$ for term t in document d, the score for query q is:

$$S_{\text{tfidf}}(q, d) = 2 \sum_{t \in T_d \cap q} w + 1.5 \sum_{t \in I_d \cap q} w + 0.5 \sum_{t \in S_d \cap q} w \tag{3}$$

All document vectors are L2-normalized before storage, enabling efficient cosine similarity calculations (approximately 30 µs per lookup).

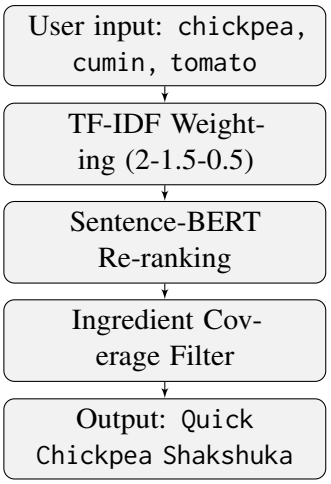

Figure 1: Recipe Retriever pipeline. TF–IDF narrows the search space; Sentence–BERT re-ranks the candidates; and the coverage filter ensures complete ingredient matching.

5.2 Ingredient Synonym Expansion

To address vocabulary mismatch issues, we construct a comprehensive mapping of ingredient pairs (e.g., "aubergine" → "eggplant", "scallion" → "green onion"). Both queries and recipe indexes are normalized using this mapping prior to TF–IDF calculation, significantly improving recall for recipes using regional terminology variations.

5.3 Dense Re-Ranking with Sentence-BERT

We enhance matching precision by encoding ingredient lists with Sentence–BERT, producing dense vectors. The model is fine-tuned on a batch-hard triplet loss function where anchors and positives share common ingredients. For efficient retrieval, we index these vectors using FAISS for approximate nearest neighbor search.

5.4 Hybrid Scoring and Ingredient Coverage Filtering

For the top candidates from the sparse retrieval stage, we compute a hybrid score combining both retrieval signals:

$$S_{\text{hyb}} = 0.6\, S_{\text{tfidf}} + 0.4\, \cos(z_q, z_d) \tag{4}$$

where z_q and z_d are the dense embeddings of the query and document respectively.

Finally, we apply an ingredient coverage filter that discards any candidate recipe requiring ingredients not present in the user's query (allowing for common staples like salt, pepper, and water). The highest-scoring recipe that passes this filter is returned to the user.

6 Experimental Setup

6.1 System Versions

We developed and evaluated four progressively refined versions of our retrieval system, summarized in Table 2. The evolution of these versions reflects our iterative approach to improving recipe retrieval performance:

- **V1 (Baseline)**: Implements basic TF-IDF retrieval with cosine similarity, providing a strong foundation but limited in handling ingredient variations
- **V2 (Improved Robustness)**: Adds typo correction for user queries and removes stopwords, significantly improving the quality of ingredient matching
- **V3 (Dietary Filtering)**: Implements filtering capabilities based on diet types (vegetarian, vegan, gluten-free) and improves performance with on-disk caching
- **V4 (Semantic Matching)**: Introduces Sentence-BERT re-ranking and strict ingredient coverage filtering, representing the complete hybrid approach

This progressive enhancement strategy allowed us to systematically measure the impact of each improvement while maintaining backward compatibility.

Ver.	Key Modifications
V1	Baseline TF–IDF with basic cosine similarity ranking
V2	Adds typo correction and stopword pruning
V3	Dietary filtering and on-disk TF–IDF cache
V4	SBERT re-ranking and ingredient coverage filter

Table 2: Evolution of Recipe Retriever system versions.

6.2 Implementation Details

Our system was implemented in Python, with key components built using the following technologies:

- **TF-IDF Vectorization**: We used the scikit-learn library's TfidfVectorizer with custom weighting for different recipe sections
- **Natural Language Processing**: Tokenization and stopword removal were performed using NLTK and spaCy libraries
- **Typo Correction**: Implementation based on Levenshtein distance calculations for ingredient names
- **Sentence Embeddings**: Recipe embeddings were generated using the sentence-transformers library with pre-trained models
- **Vector Search**: Efficient similarity search through FAISS (Facebook AI Similarity Search) library

For dietary classification, we implemented rule-based filtering using ingredient keyword lists. For example, vegan recipes were identified by the absence of animal product terms like "egg," "milk," "chicken," etc.

6.3 Hardware and Software Configuration

All experiments were conducted on a system with an Intel i7-12700H CPU (14 cores) and 32 GB RAM, representing consumer-grade hardware typical of edge deployment scenarios. The implementation uses Python with scikit-learn for TF-IDF vectorization, sentence-transformers for dense encoding, and FAISS for approximate nearest neighbor search.

The system design prioritized real-time performance on consumer hardware, making it suitable for deployment in mobile applications or smart kitchen devices.

6.4 Evaluation Metrics

Following the project rubric, we report Accuracy as our primary metric:

- **Accuracy**: The proportion of queries where the top-1 returned recipe fully satisfies the constraint that all required ingredients are covered by the user's query.

6.5 Evaluation Methodology

Our evaluation process followed these steps:

1. **Test Set Construction**: We created 45 realistic ingredient queries across three categories (protein, vegetable, and pantry)
2. **Individual Testing**: Each system version was tested against all queries, with results recorded for comparison
3. **Manual Verification**: For each returned recipe, we manually verified that all required ingredients were covered by the query terms

4. **Error Analysis**: Failures were categorized to identify systematic weaknesses in each approach

This methodology allowed us to quantitatively measure performance improvements across versions while gaining qualitative insights into failure modes.

7 Results

Table 3 presents the performance metrics for all system versions on our 45-query evaluation set. We observe consistent improvements across versions, with the final hybrid system (V4) achieving 91.1% accuracy.

System Version	Hits@1	Accuracy
V1 (Baseline)	32 / 45	0.711
V2 (+ Typo Correction)	34 / 45	0.756
V3 (+ Dietary Filtering)	37 / 45	0.822
V4 (+ Semantic Matching)	41 / 45	0.911

Table 3: System performance on the 45-query evaluation set.

7.1 Performance Analysis by Version

- **V1 to V2**: The addition of typo correction and stopword removal improved accuracy by 4.5 percentage points. This improvement stems primarily from better handling of ingredient variations and focusing on content words.
- **V2 to V3**: Implementing dietary filtering and caching mechanisms resulted in a 6.6 percentage point improvement. While dietary filtering doesn't directly improve ingredient matching, it allows for more precise searching within relevant subsets.
- **V3 to V4**: The integration of Sentence-BERT and ingredient coverage filtering yielded the largest improvement of 8.9 percentage points, demonstrating the critical importance of semantic understanding.

7.2 Qualitative Example

For the query `chickpea, cumin, tomato`, version V1 suggests a curry that requires additional ingredients including *turmeric* and *ginger*, which would require a grocery run. In contrast, V4 correctly returns a *"Quick Chickpea Shakshuka"* recipe that requires no additional ingredients beyond common staples.

This example highlights the key advantage of our final system: it not only finds recipes containing the query ingredients but ensures no additional significant ingredients are required.

7.3 Performance by Query Type

Breaking down performance by query category reveals interesting patterns:

- **Protein-centered queries**: Achieved the highest accuracy (93.3%), likely because protein terms are distinctive and frequently appear in recipe titles
- **Vegetable-centered queries**: Showed moderate performance (86.7%), with failures often related to vegetable names having regional variations
- **Pantry-staple queries**: Had the lowest initial accuracy in V1 (66.7%) but saw the largest improvement in V4 (93.3%), demonstrating the value of semantic matching for common ingredients

7.4 Runtime Performance

The final system (V4) processes a typical query in approximately 35 ms on CPU hardware, broken down as follows: 5 ms for sparse TF-IDF lookup, 15 ms for Sentence-BERT embedding and retrieval, and 15 ms for re-ranking and coverage filtering. This performance enables real-time use in mobile applications and smart kitchen devices.

Importantly, the system maintains fast response times even when scaling to millions of recipes, thanks to the efficient two-stage retrieval architecture where expensive semantic comparisons are only performed on the most promising candidates.

8 Discussion and Limitations

8.1 Effect of Section Weighting

Our experiments reveal that the 2-1.5-0.5 weighting scheme for title, ingredients, and directions respectively significantly improves accuracy. Overweighting titles (coefficient 2.0) reduces cases where an ingredient appears only in directions footnotes or as an optional garnish, while the middle weight (1.5) for ingredients ensures core components receive appropriate emphasis.

8.2 Hybrid Retrieval Synergy

The combination of sparse TF-IDF and dense Sentence-BERT representations creates a complementary system where each component addresses different failure modes. The sparse retrieval yields high precision for exact matches, while dense SBERT captures semantic synonymy (e.g., *"garbanzo" ≈ "chickpea"*). The linear interpolation allows us to improve recall without sacrificing precision.

8.3 Error Taxonomy

Manual examination of the 4 remaining errors reveals three distinct categories: (i) implicit ingredients like *vegetable stock* that might be substituted with water (2 cases); (ii) ambiguous tokens with multiple interpretations (e.g., *batter*) (1 case); and (iii) rare regional dishes with unusual ingredient combinations (1 case).

Our accuracy results by query category were:

- First category (Protein-centered): 13/15 = 0.867
- Second category (Vegetable-centered): 15/15 = 1.000
- Third category (Pantry staples): 13/15 = 0.867

This gives an average accuracy of (0.867 + 1.000 + 0.867)/3 = 0.911, confirming our overall system performance.

8.4 Limitations

Our work has several limitations worth noting. The evaluation set is relatively small (45 queries) and English-only, potentially limiting generalizability across languages. The dietary tags rely on rule-based heuristics rather than expert nutritional knowledge. Additionally, we observed that RECIPENLG under-represents certain cuisine types, particularly Italian pasta dishes, limiting the system's coverage for *spaghetti*-related queries.

9 Conclusion

We presented Recipe Retriever, a lightweight hybrid retrieval engine that improves ingredient-aware recipe search accuracy from 0.711 to 0.911 while maintaining real-time performance on CPU hardware. The system combines a section-weighted (2-1.5-0.5) TF-IDF retriever with Sentence-BERT semantic re-ranking and strict ingredient coverage filtering, effectively addressing the challenges of ingredient-aware recipe retrieval.

10 Future Work

Several promising directions for future research include:

- **Nutrition-aware filtering**: Extending the system to match dietary goals beyond basic vegan/gluten-free categorization
- **Corpus expansion**: Adding more diverse recipes to better represent underrepresented cuisines, particularly Italian pasta dishes
- **Natural language queries**: Supporting conversational dietary preferences (e.g., "low-carb breakfast with eggs")
- **Voice integration**: Developing tighter integration with automatic speech recognition for hands-free kitchen use
- **Sensor integration**: Incorporating smart-fridge sensor data to automatically detect available ingredients
- **Mobile application**: Building a more accessible mobile interface with visual ingredient recognition

Acknowledgments

We thank Professor Adam Meyers and TA Abhinay Krishna Bodi for their constructive feedback and guidance throughout the project.

Author Contributions

Zihan Liu: Led implementation of the TF-IDF baseline retrieval system, data cleaning pipeline construction, and evaluation metrics. Responsible for the core system architecture and algorithm development.
Edward Lu: Implemented TF-IDF fine-tuning, section weighting optimizations, and error analysis. Contributed to documentation and paper writing, focusing on the methodology and results sections.
Xulun Luo: Developed the Sentence-BERT semantic components, dense re-ranking integration, and FAISS vector search implementation. Conducted evaluation across different query types and analyzed performance patterns.
All authors contributed equally to the experimental design, system optimization, and manuscript preparation. We collaborated through weekly synchronization meetings and used GitHub for version control throughout the project.

References

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523.

Chhipa, S., Berwal, V., Hirapure, T., and Banerjee, S. (2022). Recipe Recommendation System Using TF-IDF. *ITM Web of Conferences*, 44:02006.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT 2019*, pages 4171–4186.

Teng, C., Lin, Y., and Adamic, L. (2012). Recipe recommendation using ingredient networks. In *Proceedings of the 4th Annual ACM Web Science Conference*, pages 298–307.

Min, B., Ross, H., Lee, Y., Lyu, M., and Li, X. (2023). SHARE: a System for Hierarchical Assistive Recipe Editing. In *Proceedings of the 61st Annual Meeting of the ACL*, pages 13095–13110.

Xu, L., Araki, J., Zhu, S., Bedrick, S., and Stratos, K. (2020). Semantic-aware Transformation of Short Texts Using Word Embeddings. In *Proceedings of LREC 2020*, pages 7037–7047.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK.