RecipeOnt Chatbot (Cosys IP Report)

Vaibhav Chopra Paarth Goyal Utkarsh Dhilliwal
Indraprastha Institute of Information Technology, Delhi
vaibhav22552@iiitd.ac.in, paarth22343@iiitd.ac.in, utkarsh22551@iiitd.ac.in

1. Introduction

RecipeOnt is a comprehensive and structured ontology that blends traditional culinary wisdom with modern food science. By integrating data from RecipeDB and FlavorDB, it constructs a rich knowledge graph encompassing 118,000 recipes, molecular flavor compositions, nutritional information, and detailed cooking processes. RecipeOnt surpasses existing efforts by mapping ingredient relationships, step-by-step preparation methods, and cultural contexts in a user-friendly and scalable format. Moreover, it adheres to global standards such as schema.org and rdfs, ensuring seamless integration with other systems and applications.

The primary objective of our project is to leverage a knowledge graph containing multiple RDF triple statements constructed using the RecipeOnt ontology to enhance the capabilities of a Large Language Model (LLM). By integrating structured knowledge from RecipeOnt which includes detailed information on recipes, ingredients, flavor moleclues, flavor profiles, and their interrelationships. The chatbot is able to access precise and context-rich information beyond the limitations of typical LLMs. This augmentation enables the chatbot to provide accurate, context-aware answers that reflect real recipe data and preparation steps from the underlying databases. Ultimately, the project aims to develop an intelligent conversational system that is not only knowledgeable about culinary concepts but also capable of delivering personalized and reliable guidance grounded in a comprehensive food knowledge graph.

2. Knowledge Graph Structure

The core entities and their relationships are illustrated in figure-1. Although the ontology defines a wide range of entities and relationships such as ingredients, recipes, cooking methods, utensils, nutritional attributes, and cultural associations, not all of these are fully represented in the actual knowledge graph.

The actual knowledge graph is based upon two main entities i.e. recipe and ingredient. A recipe has it's corresponding ID, mapping to constituent ingredients, recipe description ID which is further mapped to the actual textual description and region of origin (refer to Figure 2). While an ingredient has mappings to it's constituent flavor molecules which are then in turn mapped to the actual taste conferred by that molecules (for example floral, burnt, phenolic etc.)

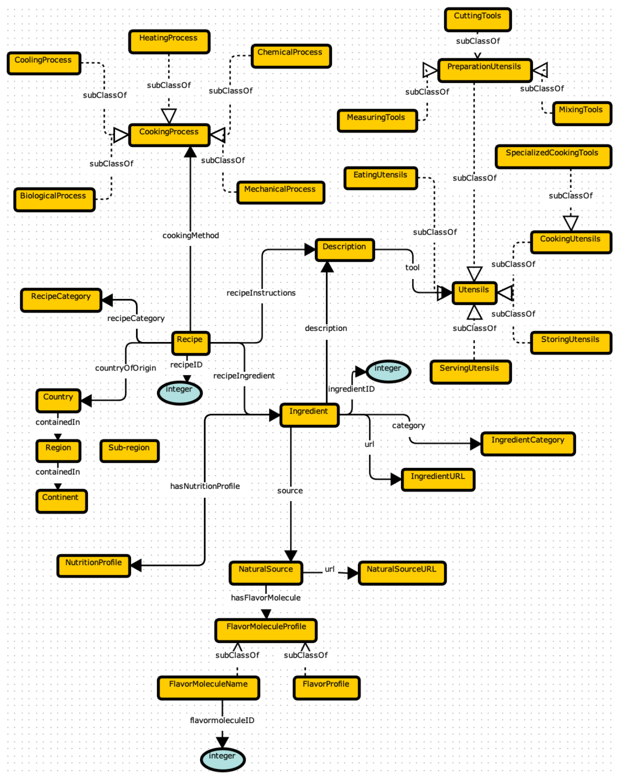
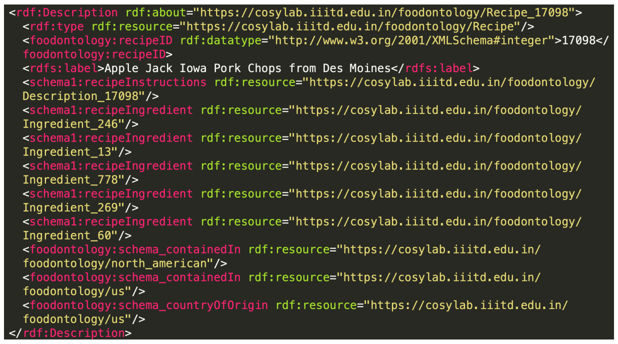


Figure 1. Ontology Structure

Please note that even though the ID's for each ingredient are contiguous starting from 1, there are some ingredients which do not have their detailed mapping to flavor molecules as shown in Figure 3.

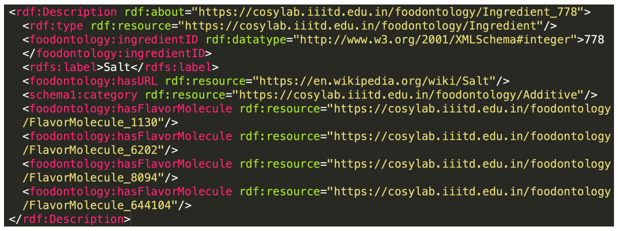
3. Methodology

To build the vector database for recipes and ingredients, the code begins by loading the knowledge graph, which is an RDF file generated using the RecipeOnt



```
<rdf:Description rdf:about="https://cosylab.iiitd.edu.in/foodontology/Recipe_17098">
  <rdf:type rdf:resource="https://cosylab.iiitd.edu.in/foodontology/Recipe"/>
  <foodontology:recipeID rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">17098</
  foodontology:recipeID>
  <rdfs:label>Apple Jack Iowa Pork Chops from Des Moines</rdfs:label>
  <schema1:recipeInstructions rdf:resource="https://cosylab.iiitd.edu.in/foodontology/
  Description_17098"/>
  <schema1:recipeIngredient rdf:resource="https://cosylab.iiitd.edu.in/foodontology/
  Ingredient_246"/>
  <schema1:recipeIngredient rdf:resource="https://cosylab.iiitd.edu.in/foodontology/
  Ingredient_13"/>
  <schema1:recipeIngredient rdf:resource="https://cosylab.iiitd.edu.in/foodontology/
  Ingredient_778"/>
  <schema1:recipeIngredient rdf:resource="https://cosylab.iiitd.edu.in/foodontology/
  Ingredient_269"/>
  <schema1:recipeIngredient rdf:resource="https://cosylab.iiitd.edu.in/foodontology/
  Ingredient_60"/>
  <foodontology:schema_containedIn rdf:resource="https://cosylab.iiitd.edu.in/
  foodontology/north_american"/>
  <foodontology:schema_containedIn rdf:resource="https://cosylab.iiitd.edu.in/
  foodontology/us"/>
  <foodontology:schema_countryOfOrigin rdf:resource="https://cosylab.iiitd.edu.in/
  foodontology/us"/>
</rdf:Description>
```

Figure 2. Singular Recipe in Knowledge Graph



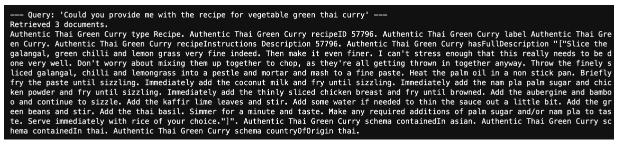
```
<rdf:Description rdf:about="https://cosylab.iiitd.edu.in/foodontology/Ingredient_778">
  <rdf:type rdf:resource="https://cosylab.iiitd.edu.in/foodontology/Ingredient"/>
  <foodontology:ingredientID rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">778
  </foodontology:ingredientID>
  <rdfs:label>Salt</rdfs:label>
  <foodontology:hasURL rdf:resource="https://en.wikipedia.org/wiki/Salt"/>
  <schema1:category rdf:resource="https://cosylab.iiitd.edu.in/foodontology/Additive"/>
  <foodontology:hasFlavorMolecule rdf:resource="https://cosylab.iiitd.edu.in/foodontology
  /FlavorMolecule_1130"/>
  <foodontology:hasFlavorMolecule rdf:resource="https://cosylab.iiitd.edu.in/foodontology
  /FlavorMolecule_6202"/>
  <foodontology:hasFlavorMolecule rdf:resource="https://cosylab.iiitd.edu.in/foodontology
  /FlavorMolecule_8094"/>
  <foodontology:hasFlavorMolecule rdf:resource="https://cosylab.iiitd.edu.in/foodontology
  /FlavorMolecule_644104"/>
</rdf:Description>
```

Figure 3. Singular Ingredient in Knowledge Graph

ontology. This ontology defines concepts such as recipes, ingredients, and their properties, and the file includes real instances derived from databases like RecipeDB and FlavorDB. The rdflib library is used to parse this RDF file, allowing the program to query entities and their relationships in a structured manner.

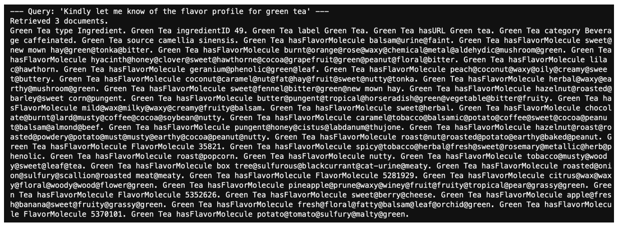
The next step focuses on identifying recipe and ingredient entities within the graph. Recipes are detected by checking for entities that are labeled as type DBO.Recipe, while ingredient instances are identified using the presence of the property DBO.ingredientID. Once these entities are found, the goal is to convert their structured representation into natural language-like text documents that an embedding model can understand. For each recipe or ingredient entity, the code collects all related triples in the form of subject, predicate, and object. These triples are then converted into human-readable sentences such as “Pasta containsIngredient Tomato”. To improve readability, the code uses a helper function to replace URIs with their label or a cleaned name if a label is unavailable. Also note that the recipe is mapped to ingredientIDs and a descriptionID which is of no use to us directly, it is ensured that the code recursively moves in depth and resolves all such IDs and replaced them with the actual content (for example the actual textual description)

For recipes, additional descriptive information such as detailed cooking instructions is extracted when available. This content is also added to the text document to enrich its contextual meaning. Similarly, ingredient documents are created by converting their RDF triples into textual descriptions. The result is a collection of text-based documents, where each document represents either a recipe or an ingredient, describing its properties, relationships, and attributes in sentence form.



```
--- Query: 'Could you provide me with the recipe for vegetable green thai curry' ---
Retrieved 3 documents.
Authentic Thai Green Curry type Recipe. Authentic Thai Green Curry recipeID 57796. Authentic Thai Green Curry label Authentic Thai Gre
en Curry. Authentic Thai Green Curry recipeInstructions Description 57796. Authentic Thai Green Curry hasFullDescription "["Slice the
galangal, green chilli and lemon grass very fine indeed. Then make it even finer. I can't stress enough that this really needs to be d
one very well. Don't worry about mixing them up together to chop, as they're all getting thrown in together anyway. Throw the finely s
liced galangal, chilli and lemongrass into a pestle and mortar and mash to a fine paste. Heat the palm oil in a non stick pan. Briefly
fry the paste until sizzling. Immediately add the coconut milk and fry until sizzling. Immediately add the nam pla palm sugar and chic
ken powder and fry until sizzling. Immediately add the thinly sliced chicken breast and fry until browned. Add the aubergine and bambo
o and continue to sizzle. Add the kaffir lime leaves and stir. Add some water if needed to thin the sauce out a little bit. Add the gr
een beans and stir. Add the thai basil. Simmer for a minute and taste. Make any required additions of palm sugar and/or nam pla to tas
te. Serve immediately with rice of your choice."]". Authentic Thai Green Curry schema containedIn asian. Authentic Thai Green Curry sc
hema containedIn thai. Authentic Thai Green Curry schema countryOfOrigin thai.
```

Figure 4. Recipe document which will be embedded and provided to LLM as context



```
--- Query: 'Kindly let me know of the flavor profile for green tea' ---
Retrieved 3 documents.
Green Tea type Ingredient. Green Tea ingredientID 49. Green Tea label Green Tea. Green Tea hasURL Green tea. Green Tea category Bevera
ge caffeinated. Green Tea source camellia sinensis. Green Tea hasFlavorMolecule balsam@urine@faint. Green Tea hasFlavorMolecule sweet@
new mown hay@green@tonka@bitter. Green Tea hasFlavorMolecule burnt@orange@rose@waxy@chemical@metal@aldehydic@mushroom@green. Green Tea
hasFlavorMolecule hyacinth@honey@clover@sweet@hawthorne@cocoa@grapefruit@green@peanut@floral@bitter. Green Tea hasFlavorMolecule lila
c@hawthorn. Green Tea hasFlavorMolecule geranium@phenolic@green@leaf. Green Tea hasFlavorMolecule peach@coconut@waxy@oily@creamy@swee
t@buttery. Green Tea hasFlavorMolecule coconut@caramel@nut@fat@hay@fruit@sweet@nutty@tonka. Green Tea hasFlavorMolecule herbal@waxy@ea
rthy@mushroom@green. Green Tea hasFlavorMolecule sweet@fennel@bitter@green@new mown hay. Green Tea hasFlavorMolecule hazelnut@roasted@
barley@sweet corn@pungent. Green Tea hasFlavorMolecule butter@pungent@tropical@horseradish@green@vegetable@bitter@fruity. Green Tea ha
sFlavorMolecule mild@wax@milky@waxy@creamy@fruity@balsam. Green Tea hasFlavorMolecule sweet@herbal. Green Tea hasFlavorMolecule chocol
ate@burnt@lard@musty@coffee@cocoa@soybean@nutty. Green Tea hasFlavorMolecule caramel@tobacco@balsamic@potato@coffee@sweet@cocoa@peanu
t@balsam@almond@beef. Green Tea hasFlavorMolecule pungent@honey@cistus@labdanum@thujone. Green Tea hasFlavorMolecule hazelnut@roast@ro
asted@powdery@potato@must@musty@earthy@cocoa@peanut@nutty. Green Tea hasFlavorMolecule roast@nut@roasted@potato@earthy@baked@peanut. G
reen Tea hasFlavorMolecule FlavorMolecule 35821. Green Tea hasFlavorMolecule spicy@tobacco@herbal@fresh@sweet@rosemary@metallic@herb@p
henolic. Green Tea hasFlavorMolecule roast@popcorn. Green Tea hasFlavorMolecule nutty. Green Tea hasFlavorMolecule tobacco@musty@wood
y@sweet@leaf@tea. Green Tea hasFlavorMolecule box tree@sulfurous@blackcurrant@cat-urine@meaty. Green Tea hasFlavorMolecule roasted@oni
on@sulfury@scallion@roasted meat@meaty. Green Tea hasFlavorMolecule FlavorMolecule 5281929. Green Tea hasFlavorMolecule citrus@wax@wax
y@floral@woody@wood@flower@green. Green Tea hasFlavorMolecule pineapple@prune@waxy@winey@fruit@fruity@tropical@pear@grassy@green. Gree
n Tea hasFlavorMolecule FlavorMolecule 5352626. Green Tea hasFlavorMolecule sweet@berry@cheese. Green Tea hasFlavorMolecule apple@fres
h@banana@sweet@fruity@grassy@green. Green Tea hasFlavorMolecule fresh@floral@fatty@balsam@leaf@orchid@green. Green Tea hasFlavorMolecu
le FlavorMolecule 5370101. Green Tea hasFlavorMolecule potato@tomato@sulfury@malty@green.
```

Figure 5. Ingredient document which will be embedded and provided to LLM as context

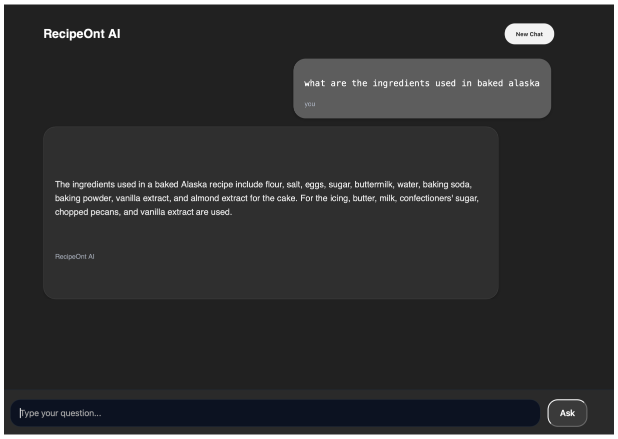
Once the textual representation is prepared, these documents are converted into numerical vector embeddings using the SentenceTransformer model (all-MiniLM-L6-v2). This model is capable of transforming text into high-dimensional vectors that capture semantic meaning. Embeddings are generated in batches to handle large data efficiently. These embeddings enable the system to compare documents based on meaning rather than exact keyword matches.

After the embeddings are created, they are indexed using FAISS, a fast similarity search library. FAISS organizes the vectors in a way that supports efficient retrieval of semantically similar documents. This forms the core of the vector database, where both the original text documents and their corresponding vector representations are stored. The index is then saved to disk along with the documents, making it reusable for retrieval tasks.

Finally, when a user submits a query, the system converts the query into an embedding using the same SentenceTransformer model. It then searches the FAISS index to find the most similar documents based on semantic meaning. These retrieved documents are used as contextual knowledge and passed to a language model to generate an answer. This allows the system to provide responses that are grounded in real data from the knowledge graph, making it both context-aware and factually relevant.

The chatbot is deployed through a web-based frontend built using the Flask framework, providing users with an intuitive and interactive interface to ask culinary-related queries. When a user submits a question, it is processed by our retrieval mechanism, which fetches the most relevant context from the knowledge graph-based vector database. This contextual information is then supplied to the locally hosted Mistral 7B language model, which is run via Ollama.

4. How To Run The Chatbot Locally



what are the ingredients used in baked alaska

you

The ingredients used in a baked Alaska recipe include flour, salt, eggs, sugar, buttermilk, water, baking soda, baking powder, vanilla extract, and almond extract for the cake. For the icing, butter, milk, confectioners' sugar, chopped pecans, and vanilla extract are used.

RecipeOnt AI

Figure 6. Chatbot frontend

To run the chatbot locally, begin by ensuring that Ollama is installed on your system, as it is required to locally host and serve the language model. Next, download the desired model such as the Mistral 7B parameter model and place it in the project directory. Open a terminal in the same directory and execute the command ollama create mistral7b -f Modelfile to set up the model with Ollama. After that, install all required Python libraries by manually installing the dependencies mentioned in the code. Once all setups are complete, simply run the Flask application by executing the relevant Python file i.e. app.py. This will launch the chatbot locally, allowing access through the web based interface in your browser.

5. Model Evaluation

G-Eval (Generative Evaluation) is a method that uses large language models (LLMs) to automatically evaluate the quality of responses generated by other LLMs based on natural language criteria such as relevance, factual accuracy, clarity, coherence, helpfulness, and completeness. Instead of relying on traditional metrics like BLEU or ROUGE, which focus mainly on text overlap, G-Eval prompts an LLM (acting as a judge) to assess a model's output using human like reasoning and provide both a score and an explanation. This makes it scalable, flexible, and closer to human evaluation, making it useful for tasks like comparing LLMs, fine-tuning models, grading open-ended answers, and monitoring chatbot quality.

6. Conclusion and Future Improvements

The Retrieval-Augmented Generation (RAG) approach significantly enhances the accuracy and reliability of chatbot responses by combining the strengths of information retrieval and generative language models. Traditional language models, while powerful, often suffer from issues such as hallucination and lack of factual grounding, especially when dealing with specialized domains like culinary science. By integrating a retrieval mechanism that accesses a knowledge graph-based vector database, the chatbot is able to fetch factual and context-specific information regarding recipes, ingredients, nutritional attributes, and preparation steps. This ensures that the responses are not only fluent and coherent but also grounded in verified knowledge extracted from RecipeOnt. Moreover, the RAG framework enables the system to remain scalable, adaptable, and explainable. The knowledge graph can be continuously expanded with new recipes, ingredients, or cultural associations without retraining the language model, making the system future-proof and easily maintainable. Additionally, because the responses are generated based on retrieved context, it becomes easier to trace back the source of information, improving transparency and trustworthiness. This combination of semantic retrieval and knowledge-grounded generation makes RAG an ideal approach for building intelligent, domain-aware conversational systems like our recipe chatbot.

- To further enhance response speed and improve overall chatbot performance, several optimizations can be implemented, including using caching to avoid repeated retrieval for common queries and optimizing the FAISS index through GPU acceleration. Additionally, replacing the current model with more advanced open-source LLMs such as LLaMA 3, Mistral Large, DeepSeek, or Qwen can significantly improve reasoning, contextual understanding, and generation quality. These models not only offer better performance but also support efficient inference, making them well suited for real time RAG-based applications where both speed and accuracy are essential.

- To improve response quality when dealing with ingredient flavor profiles, it is important to structure and filter the information before passing it to the LLM. Since a single ingredient may be linked to multiple flavor molecules and each molecule may further contain multiple flavor notes, directly feeding all of this information may lead to information overload and reduce answer relevance. Instead, the system can preprocess and summarize flavor data by grouping similar flavor molecules, ranking them based on their sensory significance, and filtering out low impact or redundant flavor notes. Additionally, clustering techniques can be used to categorize flavor notes into broader groups such as fruity, earthy, nutty, or spicy, reducing granularity while preserving meaning. By providing the LLM with structured, concise, and hierarchically organized flavor information, we can minimize noise and enhance response clarity, allowing the chatbot to deliver more accurate, human friendly explanations of ingredient flavor profiles.

- Using the generative capabilities of modern LLMs, the system can go beyond factual retrieval and leverage the structured information in the knowledge graph for creative tasks such as personalized recipe generation. By combining user preferences such as dietary restrictions, cuisine type, available ingredients, and desired flavor profiles with structured data on ingredient pairings, flavor molecules, and cultural context from the knowledge graph, an LLM can intelligently compose novel recipes that are both feasible and gastronomically meaningful. The knowledge graph provides factual grounding and ensures that the generated recipes maintain ingredient compatibility, appropriate cooking methods, and flavor coherence, while the LLM contributes creativity, natural language expression, and adaptability to user needs. This hybrid approach enables the chatbot to not only answer questions but also act as a culinary assistant capable of generating customized, flavor-aware recipes tailored to individual tastes.

- Kindly coordinate with the original creators of RecipeOnt and the knowledge graph in order to obtain the latest updated version (devyansh22156@iiitd.ac.in, daksh21459@iiitd.ac.in and mansig@iiitd.ac.in). The methodology of creating and retrieving documents for recipes and ingredients can be utilised even when there are updations in the knowledge graph.