# Bundle Recommender from Recipes to Shopping Carts - Optimizing Ingredients, Kitchen Gadgets and their Quantities

Chahak Sethi[1,†], Melvin Vellera[1,†], Diane Myung-kyung Woodbridge[1] and Joey Jonghoon Ahnn[2]

[1]University of San Francisco, San Francisco, California, USA
[2]Target, Sunnyvale, California, USA

**Abstract**

In this paper, we introduce a recommender system where it automatically captures the context of what users or guests look for and recommends a bundle of products to be added to their shopping cart. The recommendation system takes selected recipes from a user as input and recommends a shopping cart with ingredients in optimized quantities as well as any kitchen gadgets that might be necessary to efficiently prepare the recipes using neural networks. We propose a system architecture, dive deep into the individual components, and evaluate the performance of information retrieval, semantic search, and quantity optimization algorithms. Using an ensemble methodology, we attained a mean average precision of over 0.9 for ingredient and quantity recommendations. The recipe-based bundle recommender system may be used not only to improve the user's shopping experiences but also to enable and encourage them to have healthier eating habits, aiming at providing personalized product recommendations.

## 1. Introduction

Recommendation systems provide personalized recommendations for the customers using various data from customers and products to enhance customer experience and maximize the conversion rate, significantly contributing to revenue growth in the retail industry. In 2020, the recommendation system market was valued at 1.77 billion US dollars globally with a projected compound annual growth rate of 33.0% by 2028 [9].

Recommendation systems generally utilize two categories of algorithms, including Collaborative filtering-based recommendation [23][5] and content-based recommendation [16] algorithms. Collaborative filtering (CF) relies on the user's history data and matches a user *A* with a similar user *B* and recommends *A* what *B* liked. Today, most big e-commerce giants with a massive amount of data, use collaborative filtering to recommend products to their customers [4]. Content-based recommendations create a profile to characterize each item. Some of the industry applications of content-based recommendation systems include a name that suggests jobs to the users by matching their interests and skills with the features of job postings [2]. IMDb uses the information of the movies or TV shows, including genre, language, cast, director, and popularity, to recommend them to their users [28].

With the COVID-19 pandemic, there has been increased need and interest in meal preparation, including shopping and cooking. This even helped some people rediscover their liking for meal preparation and discover new recipes. However, in recommending items for cooking recipes, there are unique challenges, including: (1) Recommending in-stock grocery items with correct quantity as the recipe uses volume while the product uses weight as a measure. For example, in Figure 1, the recipe for classic french toast uses four tablespoons of unsalted butter, whereas it is sold in a 16-ounce pack; (2) Recommending kitchen gadgets using unstructured text data in directions. For example, in the figure 1, the recipe for classic french toast requires to "whisk", which implicitly indicates that the user would need a "whisker"; (3) Optimizing quantity for repeating ingredients in multiple recipes. For example, if the classic French toast recipe (Figure 1, Table 1) requires six slices of bread, while another recipe requires eight slices, which indicates a total of 1 pack of bread is needed to complete both recipes.

In this research, we developed a content-based recommendation system with natural language processing to solve the three aforementioned sub-problems: (1) the developed system parses ingredients and kitchen gadgets in the recipes corpus and recommends the most similar products from our product catalog database; (2) it parses unstructured text in the recipe direction section to recommend the kitchen gadgets required to cook a recipe; (3) Finally, it optimizes the number of products and bundles the recommendations together as a set of items that

*ORSUM@ACM RecSys 2022: 5th Workshop on Online Recommender Systems and User Modeling, jointly with the 16th ACM Conference on Recommender Systems, September 23rd, 2022, Seattle, WA, USA*

†The authors contributed equally.

csethi2@usfca.edu (C. Sethi); mvellera@usfca.edu (M. Vellera); dwoodbridge@usfca.edu (D. M. Woodbridge); joey.ahnn@target.com (J. J. Ahnn)

© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).



CEUR Workshop Proceedings (CEUR-WS.org)

CEUR-WS.org/Vol-3303/paper4.pdf

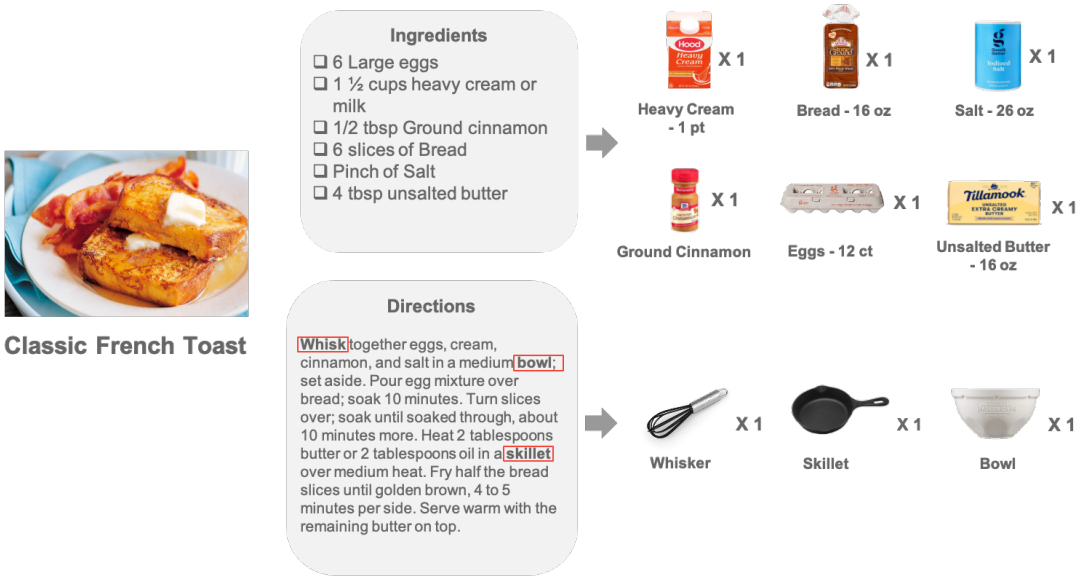


**Figure 1:** Text of an Example Recipe's (Classic French Toast) and Recommended Ingredients and Kitchen Gadgets

**Table 1**
Shopping Cart Recommendation - French Toast

| Product | Qty | Price ($) |
|---|---|---|
| Hood Heavy Cream - 1pt | 1 | 3.49 |
| McCormick Ground Cinnamon - 2.37oz | 1 | 1.99 |
| Arnold Stone Ground Wheat Bread - 16oz | 1 | 2.59 |
| Grade A Large Eggs - 12ct - Good & Gather™ | 1 | 1.69 |
| Iodized Salt - 26oz - Good & Gather™ | 1 | 0.49 |
| Tillamook Unsalted Butter Spread - 16oz | 1 | 4.19 |
| Squish 1.5qt Mixing Bowl | 1 | 8.99 |
| 9" Whisk Stainless Steel | 1 | 6.00 |
| Westinghouse Cast Iron Seasoned Skillet 6.5-inch | 1 | 23.50 |

customers can purchase individually or as a bundle, a group of complementary items that can be purchased together [13] [29].

Our research offers the convenience of personally curated shopping experiences to users by reducing their shopping efforts to find out right products as a bundle. The system helps users without any prior experience in cooking easily purchase the required ingredients and kitchen gadgets. An automatic quantity optimization will reduce wastage of resources and optimize costs for the users by suggesting the correct quantity of the product. We believe that this approach leads to increased user basket sizes, eventually raising the business revenue. Furthermore, it can also aid in automated promotional emails to the consumers with all the ingredients and gadgets in a recipe rather than the hand-curated contents which we find time-consuming.

In this paper we discuss the related work already present in this field in Section 2. Next, we provide the overview of the developed system, including mapping kitchen gadgets and ingredients in the recipes to relevant products from the product catalog database, optimizing the quantities to be recommended, and providing multiple candidates and their corresponding ranking to the user in Section 3. We have employed a few techniques to measure the performance of the system which will be discussed in Section 4. We summarize our work with some future works section 5. The authors make the codes used for the research available to the public [24].

## 2. Related Work

In late 2015, an American company that operates a grocery delivery and pick-up service in the United States and Canada, integrated with AllRecipes, a top recipe site, to allow users to select a recipe and fill their cart with all the necessary ingredients [22]. Although the ingredient recommendations provided by a e-commerce company are accurate for a good number of recipes, we observed that the recommended quantities were not ideal for certain recipes. There were also cases where no matching product was found for a recipe ingredient, such as vegetable oil, even though there are other closely related products, including olive oil and canola oil. In addition to these limitations, we also identified an opportunity for augmenting ingredient recommendations with kitchen gadget recommendations that could help improve a user's cooking experience, especially those new to cooking. Our

work has been primarily motivated by these use cases, and in this paper, we propose a methodology that generates accurate ingredient recommendations as well as kitchen gadget recommendations.

To our knowledge, there has not been any notable research regarding the recommendations of an optimized shopping cart of ingredients and kitchen gadgets based on recipes. We find that most of the existing literature in the food domain is related to recommending recipes or ingredient substitutes. [25, 26, 11, 17]. Anirudh Jagithyala [11] developed a recommendation system that recommends recipes based on recipe ratings, ingredients, and review text. A number of approaches including memory-based collaborative filtering and TF-IDF were tried along with similarity measures such as cosine and Pearson correlation. The research evaluated multiple approaches using the mean average precision (mAP) and showed that collaborative filtering on recipe ratings performed better. Chantal Pellegrini et al. [17] explored the use of text and image embeddings for identifying ingredient substitutes. They generated context-free embeddings using word2vec as well as context-based embeddings using transformer-based models. In the end, the research showed that the transformer-based multi-modal approach using text and image embeddings together gave the best results with a precision of 0.84 for the top 1000 most common ingredients. Chun-Yuen Teng et al. [25] explored the recommendation of recipes and ingredient substitutes using network structures. The system identifies ingredient substitutes using a graph structure where nodes represent ingredients and edges represent the degree of substitutability. To derive pairs of related recipes, they computed the cosine similarity between the ingredient lists for the two recipes, weighted by the inverse document frequency. Mayumi Ueda et al. [26] applied user preferences and ingredient quantity for recommending recipes. Their method breaks recipes down into their ingredients and scores them based on the ingredients' frequency of use and specificity.

Some of the existing algorithms for searching and ranking relevant items use classical information retrieval algorithms such as TF-IDF [19], BM-25 [21], or Glove [18], while others make use of deep learning models such as BERT [8]. BERT [8] is a language representation model that can give accurate contextual embeddings for words in most cases. Unfortunately, the BERT does not generally give accurate representations for sentences, and the construction of BERT makes it unsuitable for semantic similarity search. To overcome these issues, we applied the Sentence-BERT, [20] model, which was trained using Siamese BERT-Networks.

The preferred recommendation system architecture was one based a two-stage approach consisting of candidate generation and ranking stages. This two-stage approach allows for recommendations from a very large corpus (millions) while still being certain that recommendations are personalized and engaging for the user. Our research employed a two-stage approach for candidate generation (retriever stage) and ranking (re-ranker stage) [7].

## 3. System Overview

The proposed system first takes a recipe as input from a guest shopping on an e-commerce company's website. The recipe then gets split into two sections: 1) cooking instructions and 2) ingredients. The cooking instructions are parsed in order to detect and extract any kitchen gadgets that might be required for the recipe, while the required ingredients and quantities are extracted from the recipe's ingredients section. The quantities and units of ingredients in the recipe text and the product catalog database are standardized for accurately matching varying units from recipes and products. The ingredients and kitchen gadgets required by the recipe are then fed into the recommender system to search for the best matching products from the product catalog database based on textual and quantity information. The system then adds these products to the shopping cart of the guest (Figure 2).

For advanced natural language processing (NLP), we utilized Open-source software libraries, including Spacy [10] and NLTK [3] for extracting recipe ingredients and kitchen gadgets from the recipe text. The ingredients, along with the required quantities, were parsed from the ingredient section of the recipe (Figure 1) using regular expressions, after which these ingredients were then pre-processed using the NLTK library for stop words removal, stemming, and *n*-gram expansion. The Spacy library was used for extracting kitchen gadgets from the recipe instructions using named entity recognition (NER). Once the ingredients and kitchen gadgets of the recipe are identified, they are compared against the products in the product catalog database to get the most relevant product in stock for each ingredient and user. This process also involved the novel usage of a language representation model, Bidirectional Encoder Representations from Transformers (BERT) [8], which is designed to pre-train deep bidirectional representations from an unlabeled text by jointly conditioning on both left and right contexts in all layers. The advantage of using a pre-trained architecture is that we can use transfer learning to transfer the already trained features to the current data without the complexity of training heavy machine learning models [15].

We also developed an algorithm to recommend the optimal number of products in case a guest chooses more than one recipe using the same products. The quantity or weight of the common ingredients is recommended

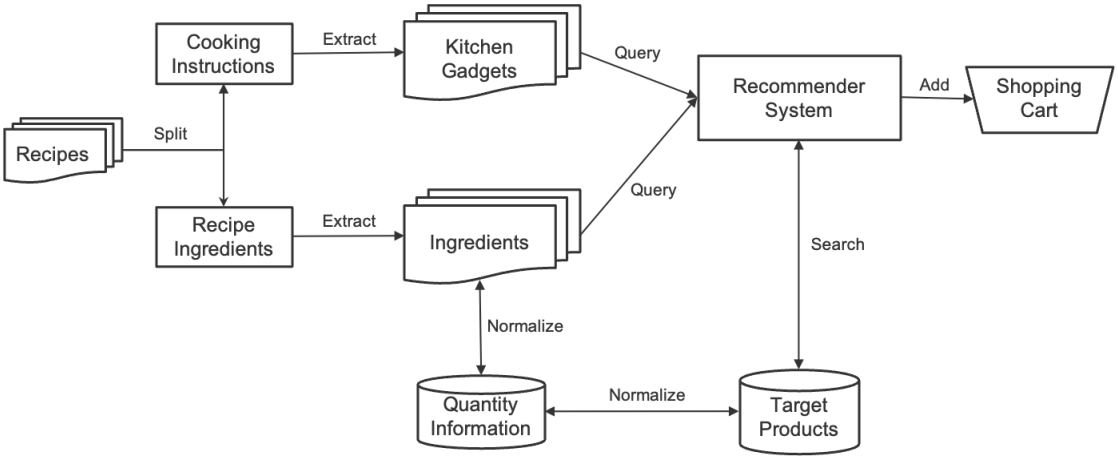


**Figure 2:** System Overview

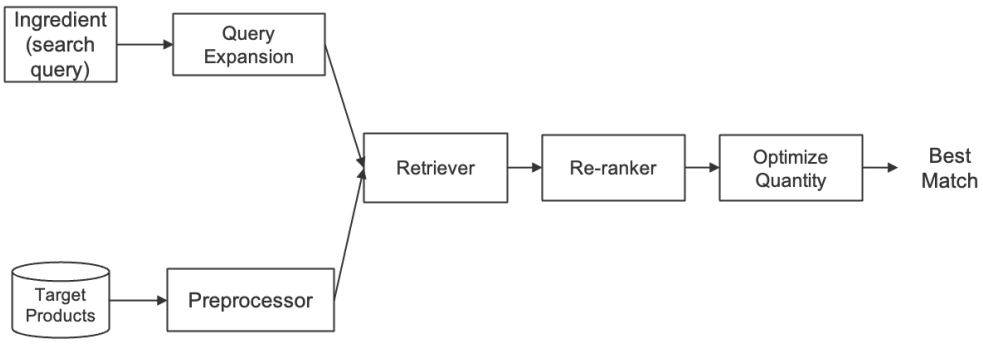


**Figure 3:** Ingredient Recommendation Workflow

based on the sum of the required amount, which helps create the optimal baskets for the guests and leads to less wastage of resources.

## 3.1. Ingredient Recommendation

The developed system utilizes a combination of semantic search and information retrieval algorithms to recommend the most relevant products for the ingredients in a recipe. Semantic search can improve search accuracy by understanding the context and content of the search query. In contrast to conventional search algorithms, which only find documents based on lexical matches, semantic search can also find synonyms. Semantic search aims to embed all entries in the corpus into vector space, where the query is embedded into the same vector space, to find the closest embeddings from the corpus. In our case, a recipe ingredient is a query, and the products are all the entries in the corpus, where the products are embedded into vector space and stored separately. During search time, the algorithm embeds a recipe ingredient into the same vector space and calculates the similarity between an ingredient ($I$) and product ($P$) to find the most relevant products. These products would have a high semantic overlap with the ingredient. We used cosine similarity in Equation 1 to find the closest embeddings from the corpus.

$$Sim(I, P) = \frac{I \cdot P}{\|I\|\|P\|} = \frac{\sum_{j=1}^{e} I_j \cdot P_j}{\sqrt{\sum_{j=1}^{e} I_j^2}\sqrt{\sum_{j=1}^{e} P_j^2}} \quad (1)$$

, where $e$ is the embedding dimension of the ingredient and product vectors.

After computing the cosine similarity between the embedding of an ingredient and the embeddings of the products, the top $m$ products are retrieved.

For complex search tasks, the search can be significantly improved by using a retrieve and re-rank framework, where the top $t$ products are retrieved efficiently, followed by a re-ranker that ranks these $t$ products and

recommends $m$ products.

#### 3.1.1. Retriever

Given an ingredient, we first use a retrieval system that quickly retrieves $t$ products that are potentially relevant for the given ingredient. Then the $t$ products are re-ranked, and the top $m$ matches are sent through to the quantity recommendation module. For our retrieval system, we make use of the BM25 algorithm for lexical search [21]. For an ingredient query ($I$) with terms $i_1, \ldots, i_n$, the BM25 score for a product text $P$ is in Equation 2.

$$BM25(P, I) = \sum_{j=1}^{n} IDF(i_j) \frac{tf(i_j, P) \cdot (c+1)}{tf(i_j) + c \cdot (1 - b + b \cdot \frac{|D|}{d_{avg}})} \tag{2}$$

, where $tf(i_j, P)$ is the number of times term $i_j$ of the ingredient text occurs in $P$. $|D|$ is the number of words in $P$. $d_{avg}$ is the average number of words for product text. $b$ and $c$ are saturation parameters for document length and term frequency respectively. In general, values such as $0.5 < b < 0.8$ and $1.2 < c < 2$ are reasonably good in many circumstances [21].

Equation 3 describes inverse document frequency ($IDF(i_j)$) for a corpus with $N$ products with term $i_j$.

$$IDF(i_j) = log \frac{N - N(i_j) + 0.5}{N(i_j) + 0.5} \tag{3}$$

, where $N(i_j)$ is the number of product texts in the database that contain the term $i_j$ of the ingredient query.

For improving the accuracy of the retrieval stage, we combined the BM25 algorithm with a bi-encoder sentence transformer model that was fine-tuned using Microsoft's MiniLM model [27]. The MiniLM model is a compressed Transformer model that uses an approach termed as deep self-attention distillation to reduce the number of parameters required by a transformer model. It is twice as fast as BERT while retaining more than 99% accuracy on SQuAD 2.0 and several GLUE benchmark tasks using only 50% of BERT's model parameters. The bi-encoder sentence transformer model [20] that uses the MiniLM model was trained using a dataset of 1 billion sentence pairs and a self-supervised contrastive learning objective: given a sentence from a sentence pair, the model should predict which out of a set of randomly sampled other sentences was actually paired with one in the dataset. A bi-encoder model performs two independent self-attentions for the query and the document, and the document is mapped to a fixed BERT representation regardless of the choice of a query. This makes it possible for bi-encoder models to pre-compute document representations offline, significantly reducing the computational load per query at the time of inference [6]. A bi-encoder model can encode an input text, such as a recipe ingredient or a product, and output a vector (embedding) that captures semantic information. If a recipe ingredient embedding and a product embedding are similar, then the cosine similarity (Equation 1) between these two embeddings will be high. Hence, by comparing an ingredient embedding with all the product embeddings using cosine similarity, we can identify the most similar products for an ingredient.

In order to reduce the search space efficiently, hierarchical classification models were created for the following levels: class, subclass, and item-type. These models were trained using the preprocessed text of the products as feature vectors and the respective hierarchical level values as the target labels. A softmax activation function (Equation 4) was used in the final layer of the multi-class classification models, along with the cross-entropy loss function (Equation 5).

$$\text{Softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)} \tag{4}$$

, where $C$ is the number of output classes, $z_i$ is an element of a vector $z$ of size $C$ corresponding to a particular class.

$$\text{Cross Entropy Loss} = -\frac{1}{N} \left( \sum_{i=1}^{N} y_i \cdot log(\hat{y}_i) \right) \tag{5}$$

, where $N$ is the number of observations, $y_i$ is the true label vector and $\hat{y}_i$ is the predicted label probability vector.

#### 3.1.2. Re-ranker

After retrieving the top $t$ products, the re-ranker stage ranks the products more accurately using a cross-encoder sentence transformer model that was fine tuned using the Microsoft's MS Marco dataset [1], which is a large scale information retrieval corpus that was created based on real user search queries using Bing search engine. In contrast to a bi-encoder model that performs two independent self-attentions for the query and the document, a cross-encoder model performs full self-attention across the entire query-document pair. As a result, the cross-encoder can model the interaction between a query and a document, and the resulting representations contain contextualized embeddings [6]. In our use case, an ingredient and a product are passed simultaneously to the cross-encoder, which then outputs a score indicating how relevant the product is for the given ingredient. As the cross-encoder models the interaction between an ingredient and a product during inference time, it is slower than the bi-encoder, and hence, it can only be used for a small subset of products. However, we can achieve a higher accuracy as they perform attention across the query and the document. After the re-ranker stage, the

**Table 2**
Commonly used units in recipes and corresponding conversion to the International System of Units (SI)

| Common Unit in Recipe | Converted Unit in International System of Units (SI) |
|---|---|
| 1 cup | 225 ml |
| 1 teaspoon | 5 ml |
| 1 tablespoon | 15 ml |
| 1 fluid ounce | 30 ml |

top $m$ matched ingredients are then optimized based on quantity.

### 3.2. Unit Normalization and Optimization

Once the algorithm selects the top $m$ matched ingredients, the next important step is to recommend the optimal quantity of the product needed in the recipe (Figure 4). For this, we start with retrieving the ingredient quantity specified in the recipe and normalize the units required to a standard SI units (Table 2), including tablespoon (tbsp), teaspoon (tsp), milliliter (ml), cup, count, pound (lb), and ounce (oz). These standard quantities are either weight or in volumes handled differently from each other.

As product descriptions generally utilize weight as a measure while most recipes use volumes, we converted the volume with the standardized unit to weight using density ($d$). For instance, the weight ($w$) for $n$ cups of a grocery product in the recipe, where 1 cup is 225 ml, can be calculated as the following.

$$w = n * 225 * d \tag{6}$$

Once the required weight is calculated, the system compares it against the weight of the recommended products in product catalog's database. The recommended number of units is then calculated for each matched product using Equation 7, where $q$ is the recommended quantity, $w$ is the ounces required in the recipe, and $p$ is the ounces sold or packaged.

$$q = \lceil \frac{w}{p} \rceil \tag{7}$$

For fresh produce items that use a count as a unit in a recipe, like two onions or three potatoes, the average weight of the given fruits and vegetables is used to convert it to weight [14]. The reverse conversion is also applied if the unit for a product at e-commerce companies uses count and the recipe specifies weight instead.

Once the recommended quantity is known for $m$ matched ingredients, we sort $m$ ingredients by the quantity recommended and the price in ascending order. In addition, the system recommends lower-priced items if multiple packaging options are available for the same products. For instance, the system recommends one pack of 1 lb rather than two packs of 0.5 lb flour if 1 lb-pack is more cost-efficient.

If a user selects multiple recipes, the quantity is optimized such that minimum units of the common products are recommended. For example, for two recipes using one tablespoon and two tablespoons of salt each, the recommended unit of salt can/bottle will be optimized to one to reduce waste and cost.

### 3.3. Kitchen Gadget Recommendation

The recommendations for kitchen gadgets follow a similar approach to ingredient recommendations using a combination of semantic search and information retrieval algorithms. The required kitchen gadget is implicitly mentioned in unstructured recipe instruction text, whereas ingredients are explicitly listed in the ingredient section. To identify the kitchen tools and methods post the pre-processing of the recipe instructions, a custom NER model from Spacy [10] was trained on these entities. The NER model identifies the kitchen gadgets (nouns) used in the recipe and the methods (verbs) that can identify a kitchen gadget. For example, for "Chop the garlic and add to the pan", a pan will be identified as a gadget, whereas chop will be identified as a method associated with the gadgets including a knife and chopping board.

Similar to ingredient recommendations, products are embedded into vector space and stored separately. During search time, a gadget from the recipe instruction is also embedded into the same vector space, and the system searches for the most relevant products. These products would have a high semantic overlap with kitchen gadgets. After computing the cosine similarity between the embedding of the kitchen gadget and the embeddings of all products, the top $m$ products with the maximum similarity are retrieved. For these search tasks, we used embeddings from the RoBERTa [12], an improved and robustly trained version of BERT with further tuned hyperparameters. RoBERTa has achieved state-of-the-art results on GLUE, RACE, and SQuAD.

For complex search tasks, the search is significantly improved by using a retrieve (Section 3.1.1 ) and re-rank (Section 3.1.1) framework, just like for the ingredients (Figure 5). For quantity optimization, if the user selects multiple recipes, the common kitchen gadgets are only recommended once, along with all the other gadgets used in each recipe.

## 4. Result

For assessing the performance of different algorithms, we identified the relevant products for the top 100 most common ingredients and kitchen gadgets (queries) and

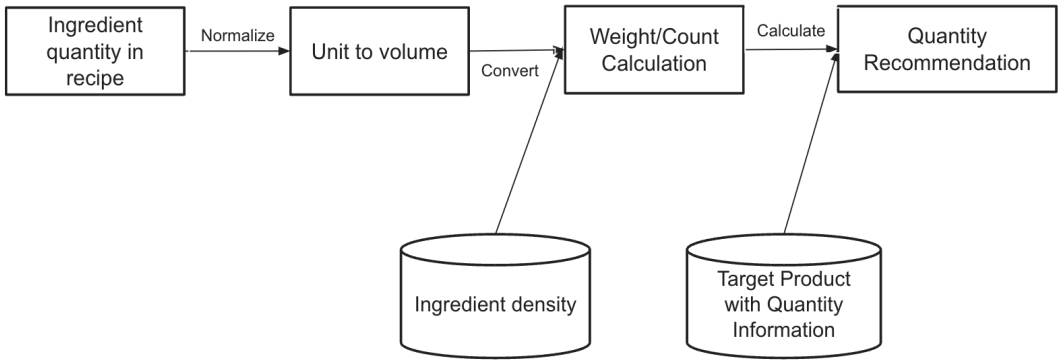


**Figure 4:** Quantity Recommendation Flow

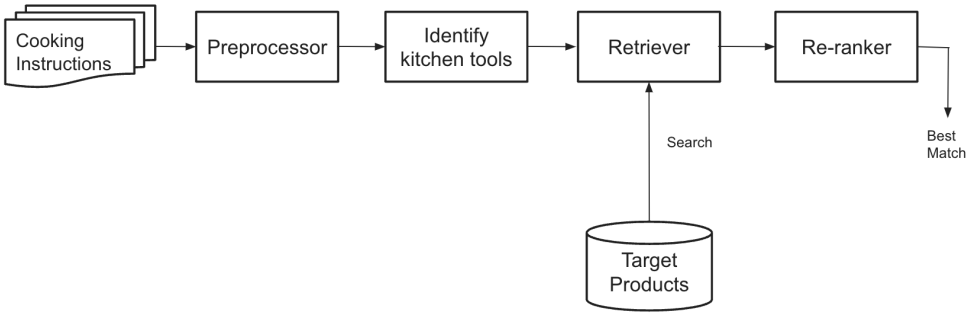


**Figure 5:** Kitchen Gadget Recommendations

calculated the mean average precision (*mAP*) at different values of $K$, where $K$ is the number of retrieved products.

$$mAP@K = \frac{\sum_{q=1}^{Q} AP@K(q)}{Q} \tag{8}$$

, where $Q$ is the number of queries, $K$ is the number of retrieved products, and $AP@K$ is the average precision at $K$.

$$AP@K = \frac{1}{min(K,R)} \sum_{k=1}^{K} P(k) \cdot rel(k) \tag{9}$$

, where $R$ is the number of relevant products, $K$ is the number of retrieved products, $rel(k)$ is an indicator that says whether that $k^{th}$ item was relevant ($rel(k)$=1) or not ($rel(k)$=0), and $P(k)$ is the precision at $k$.

$$P(k) = \frac{\sum_{i=1}^{k} rel(i)}{k} \tag{10}$$

## 4.1. Ingredient Search

For ingredient search, different algorithms, as well as an ensemble of these algorithms, were evaluated using the *mAP@K* metric. The BM25 algorithm was considered as the baseline model against more complex models. An interesting thing to note from Figure 6. is that the BM25 algorithm gives a very good performance for $K$=1 since lexical search algorithms generally have high precision due to exact keyword matching. However, for higher values of $K$, the drop in mean average precision is quite high. The experiment results showed that transformer models such as MiniLM and MS Marco are more general with consistently high precision values across different $K$ values. Using BM25 along with the MS Marco transformer model gave the best performance for all $K$ values.

Once the best performing model was identified, we further evaluated the final model with 100 recipes from the Recipe1M+ corpus, which is a large-scale, structured

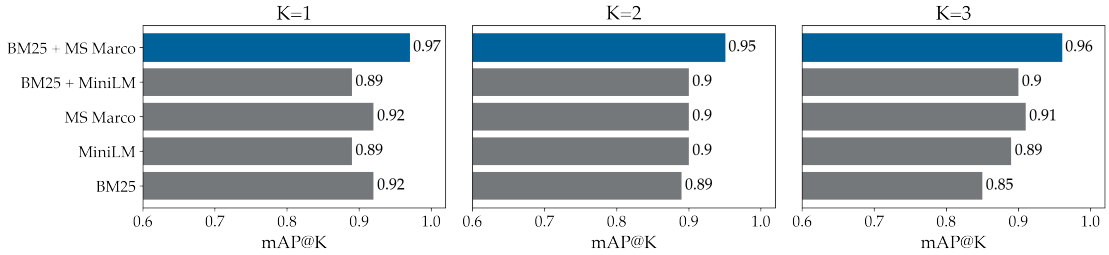


**Figure 6:** Mean Average Precision of Ingredient Search for Different $K$

**Table 3**
Quantity Recommendation Accuracy Measurement

| Ingredient | Total recipes | Correct Qty | Percentage Correct |
|---|---|---|---|
| Salt | 32,506 | 32,443 | 99.806% |
| Sugar | 19,983 | 16,473 | 82.435% |
| Butter | 18,958 | 14,875 | 78.463% |

corpus of over one million cooking recipes and 13 million food images. The *mAP*@1 value for all the ingredients from these 100 recipes was 0.949, which is similar to the *mAP@K* values we see in Figure 6 for the top 100 ingredients.

## 4.2. Quantity Normalization and Optimization

For evaluating quantity normalization, we measured the accuracy of quantity normalization, using the most commonly used ingredients in the randomly chosen 100,000 recipes. The three most commonly used ingredients, salt, sugar, and butter, are tracked to measure if the recommended quantity after unit normalization is accurate or not.

We found that salt is used in 32,506 out of the chosen 100,000 recipes ( Table 3). Out of the total 32,506 recipes, the quantity of salt is correctly recommended in 32,443 recipes which is 99.806% of the total. Similarly, the accuracies of the recommended quantity were 82.435% and 78.463% for sugar and butter respectively. However, the relatively low accuracy in quantity recommendations for sugar and butter is due to the incorrect recipe text. For example, certain recipes say 34 cups of butter instead of 3/4 cups of butter. This has been further discussed in Section 5.

Further, the quantity optimization process was also evaluated with randomly chosen 100 recipes from the Recipe1M+ corpus. The mAP@1 value for all the ingredients from these 100 recipes was 0.914. For a combination of recipes, the result has been manually verified for 5 random sets of any two recipes.

## 4.3. Kitchen Gadget

The NER model to identify kitchen gadgets was trained on 500 recipes and tested on 100 recipes. The custom entities' kitchen gadgets and methods were marked with their position in the text using regex for these 600 recipes. A manual review of the annotations was performed to update any incorrect or missed annotations, and we achieved a test F1 score of 99.627% (Equation 11).

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{11}$$

, where precision is out of the total documents retrieved, how many are relevant and recall is out of the total relevant documents how many relevant documents are retrieved.

A custom mapping is used to convert cooking methods to kitchen gadgets which along with other identified gadgets form search queries for the recipe. For evaluation, different algorithms were evaluated at mAP@K, similar to ingredient search. We developed transformer models including MiniLM, MS Marco, and Roberta for the semantic search tasks. The experiment results show that MS Marco, a combination of retriever and a re-ranker, performs consistently better than MiniLM and Roberta across all $K$ (Figure 7).

As an example, we present a recipe in Figure 8 that consists of two sections: ingredients and directions. The ingredients section is used for ingredient and quantity recommendations, whereas the directions section is used for kitchen gadget recommendations. The keywords used for kitchen gadget recommendations are highlighted in red.

The product recommendations based on the recipe (Figure 8) is given below in Table 4.

# 5. Conclusion

We presented a content-based recommendation system to recommend relevant retail products to a user or guest based on selected recipe contents. The recommended products are primarily based on the ingredients required

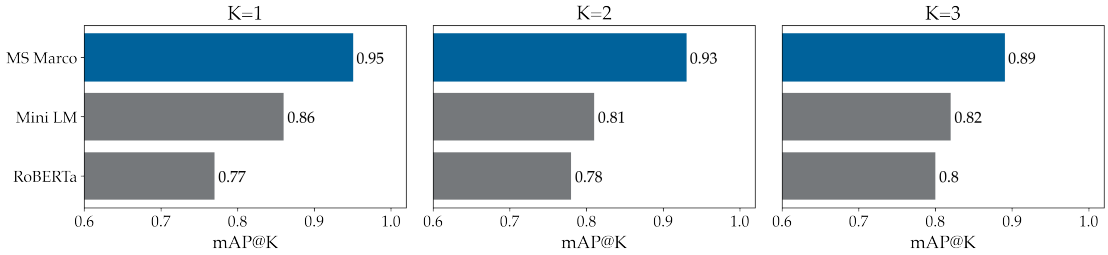


**Figure 7:** Mean Average Precision of Kitchen Gadget Search for Different $k$

**Ingredients**

1/2 cupoats
2 cupswater
2 cupspancake mix
1/2 cupapples
2 tablespoonssugar
1/2 teaspooncinnamon

**Directions**

1. In a medium bowl combine rolled oats and water, let stand 5 minutes.
2. Meanwhile, heat large nonstick skillet or griddle to medium high heat (375F).
3. Grease lightly with oil. Add remaining ingredients to rolled oats mixture; stir just until all ingredients are moistened.
4. For each pancake, pour 1/4 cup batter into hot skillet.
5. Cook 1 to 1.5 minutes, turning when edges look cooked and bubbles begin to break on surface.
6. Continue to cook 1 to 1.5 minutes or until golden brown.
7. Serve with syrup and butter, if desired.

**Figure 8:** Apple Oat Breakfast Pancakes

**Table 4**
Shopping Cart Recommendation

| Product | Qty | Price ($) |
|---|---|---|
| Red Delicious Apple | 1 | 0.99 |
| McCormick Ground Cinnamon - 2.37oz | 1 | 1.99 |
| Good & Gather Organic Oats - 18oz | 1 | 2.39 |
| Buttermilk Pancake Mix - 32oz | 1 | 2.19 |
| Imperial Granulated Pure Sugar - 4lb | 1 | 2.19 |
| Good & Gather Alkaline Water - 1L | 1 | 0.99 |
| Squish 1.5qt Mixing Bowl - Green | 1 | 8.99 |
| Anchor 8oz Glass Measuring Cup | 1 | 3.49 |
| Nylon Ladle with Soft Grip - Made By Design | 1 | 3.00 |
| Lakeside 10" Nonstick Aluminum Skillet with Faux Granite finish | 1 | 16.47 |

by the recipe, which are extracted from the recipe text along with associated quantities. More significantly, the system is also capable of recommending the relevant kitchen gadgets that might be required for making the recipe based on the instructions provided in the recipe. When a user selects multiple recipes, the recommender system optimizes quantities for each product, where the quantities are adjusted according to the amounts of the common ingredients and kitchen gadgets present in the recipes.

We conducted experiments for evaluating the effectiveness of various algorithms for the recommendation system, such as BM25, MiniLM, MS Marco, and RoBERTa. In our experiments, we compared these algorithms by assessing the top product recommendations for the 100 most frequently occurring ingredients in the 1M+ recipe corpus. For ingredient search, the ensemble approach of BM25 and MS Marco gave the best performance, while for kitchen gadget search, the MiniLM model proved to be more accurate. The accuracy of quantity recommendations was measured by evaluating certain high-frequency ingredients such as salt, sugar, and butter across more than 50,000 recipes from the 1 million+ (1M+) recipe corpus.

There were a few challenges that we faced while implementing the system that we hope to address in the near future. Firstly, the recipe text used in the 1M+ recipe corpus has some inconsistencies where the quantity required is not accurately defined. For example, a lot of recipes say 34 cups of an ingredient instead of 3/4 cups. In future work, we can work on parsing the text with more scrutiny and rules to avoid such cases. Alternatively, we could also build an entirely new recipe scraping algorithm to handle such cases. Secondly, the quantity optimization for multiple recipes is in place, but there is no formal way to measure how well is the performance of the process. As a part of future work, we would devise a way to quantify the performance of this process.

We also identified extra features for the current system. First, the dietary restrictions of a user can be accounted for by considering the ingredients, nutritional

information, and key allergens that might be present in the product using natural language processing. Second, there could be preference filters provided to users before they add a recipe to the shopping cart that could consider their dietary preferences such as whether they prefer non-GMO or organic products only. Third, instead of asking for preferences explicitly, the preferences of the users could be determined automatically by analyzing their past shopping behaviors such as clicks, views, or product purchases. Based on this implicit feedback, we could determine the user's preferences, such as whether the user is a vegetarian or if the user is currently purchasing products specific to a particular diet, such as the ketogenic diet. Such information can then be used in the recommendation system for personalizing recommended products for different users.

The content-based recommendation system proposed in this paper could potentially be used to improve the shopping experience of users by providing them options to add all the necessary products required by a recipe automatically to their shopping cart. Kitchen gadget recommendations could be helpful not just for improving the shopping experiences of the users but also for their cooking experiences. These recommendations also allow a business to increase the basket sizes of their users, thereby increasing revenue. Moreover, the system could also aid in automated promotional emails that recommend products required by recipes that may be of interest to customers.

## References


[1] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset.

[2] Shivam Bansal, Aman Srivastava, and Anuja Arora. 2017. Topic Modeling Driven Content Based Jobs Recommendation Engine for Recruitment Industry. *Procedia Computer Science* 122 (2017), 865–872. 5th International Conference on Information Technology and Quantitative Management, ITQM 2017.

[3] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

[4] Dan Chang, Hao Gui, Rui Fan, Ze Fan, and Ji Tian. 2019. Application of Improved Collaborative Filtering in the Recommendation of E-commerce Commodities. *INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS CONTROL* 14, 4 (2019), 489–502.

[5] Lin Chen, Rui Li, Yige Liu, Ruixuan Zhang, and Diane Myung-kyung Woodbridge. 2017. Machine learning-based product recommendation using Apache Spark. In *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. 1–6. https://doi.org/10.1109/UIC-ATC.2017.8397470

[6] Jaekeol Choi, Euna Jung, Jangwon Suh, and Wonjong Rhee. 2021. Improving Bi-encoder Document Ranking Models with Two Rankers and Multi-teacher Distillation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

[7] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

[9] Grand View Research. 2021. Recommendation Engine Market Size, Share & Trends Analysis. *Recommendation Engine Market Report* (2021).

[10] Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. (2017). To appear.

[11] Anirudh Jagithyala. 2014. *Recommending recipes based on ingredients and user reviews.* Ph. D. Dissertation. Kansas State University.

[12] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach.

[13] Kevin F. McCardle, Kumar Rajaram, and Christopher S. Tang. 2007. Bundling retail products: Models and analysis. *European Journal of Operational Research* 177, 2 (2007), 1197–1217.

[14] Niklas. 2022. Average Weight of All Fruits and Vegetables. https://weightofstuff.com/average-weight-of-all-fruits-and-vegetables/

[15] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (2010), 1345–1359.

[16] Michael J. Pazzani and Daniel Billsus. 2007. *Content-Based Recommendation Systems.* Springer Berlin Heidelberg, Berlin, Heidelberg, 325–341.

[17] Chantal Pellegrini, Ege Özsoy, Monika Wintergerst, and Georg Groh. 2021. Exploiting Food Embeddings for Ingredient Substitution. In *HEALTHINF*.

[18] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[19] Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, Vol. 242. Citeseer, 29–48.

[20] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.

[21] Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

[22] Perez Sarah. 2015. Instacart And Allrecipes Now Let You Add A Meal's Ingredients To Your Grocery List With A Click. https://techcrunch.com/2015/10/12/instacart-and-allrecipes-now-let-you-add-a-meals-ingredients-to-your-grocery-list-with-a-click/

[23] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. *Collaborative Filtering Recommender Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 291–324.

[24] Chahak Sethi, Melvin Vellera, Diane Myung kyung Woodbridge, and Joey Jonghoon Ahnn. 2022. Bundle Recommender from Recipes to Shopping Cart. https://github.com/dianewoodbridge/target_recipe_project/

[25] Chun-Yuen Teng, Yu-Ru Lin, and Lada A Adamic. 2012. Recipe recommendation using ingredient networks. In *Proceedings of the 4th annual ACM web science conference*. 298–307.

[26] Mayumi Ueda, Syungo Asanuma, Yusuke Miyawaki, and Shinsuke Nakajima. 2014. Recipe recommendation method by considering the users preference and ingredient quantity of target recipe. In *Proceedings of the international multiconference of engineers and computer scientists*, Vol. 1. 12–14.

[27] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. arXiv:2002.10957

[28] Chutian Wei, Xinyu Chen, Zhenning Tang, and Wen Cheng. 2021. Fully content-based IMDb movie recommendation engine with Pearson similarity. In *International Conference on Green Communication, Network, and Internet of Things (GCNIoT 2021)*, Siting Chen and Jun Mou (Eds.), Vol. 12085. International Society for Optics and Photonics, SPIE, 132 – 137.

[29] Ruiliang Yan, Chris Myers, John Wang, and Sanjoy Ghose. 2014. Bundling products to success: The influence of complementarity and advertising. *Journal of Retailing and Consumer Services* 21, 1 (2014), 48–53.