FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

FOOD SEARCH ENGINE

By

PEI-TIEN LU

A Dissertation submitted to the
Department of Statistics
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

2026

Pei-Tien Lu defended this dissertation on June 18, 2026.
The members of the supervisory committee were:

Adrian Barbu
Professor Directing Dissertation

Sonia Haiduc
University Representative

Wei Wu
Committee Member

Xufeng Niu
Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with university requirements.

ACKNOWLEDGMENTS

I wish to express my gratitude to my major professor, Dr. Adrian Barbu. His support, advice, and guidance have been foundational in my PhD studies and this dissertation. I also wish to extend my sincere appreciation to my committee members, Dr. Wei Wu, Dr. Xufeng Niu and to my university representative Dr. Sonia Haiduc. Their expertise, encouragement, and invaluable insights have significantly enhanced my research and scholarly endeavors. Lastly, I also want to thank to the Department of Statistics at Florida State University for granting me the exceptional opportunity to pursue my PhD studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

The following short list of symbols are used throughout the document. The symbols represent quantities that I tried to use consistently.

W	The input string sequence of slot filling method
w_i	The element of w
S	The slot sequence of slot filling method
s_i	The element of s
V_i	The chain of nodes of dynamic programming method
L	The values $\{0, 1, 2\}$ defining whether the nutrition value is before the item
l_i	A set of possible labels that $l_i \in L$
$\theta_k(i)$	The unary cost for node V_k taking value i
$\theta_k(i, j)$	The binary cost for node V_k taking value i and node V_{k+1} taking value j
$M_k(j)$	The min cost of the partial optimization on V_1,...,V_k with the label of V_k being j
p	The precision in evaluation measure
r	The recall in evaluation measure
F_1	The F_1-score defined as $F_1 = 2pr/(p + r)$
M	The number of correct items output by the method from the ground truth table
N	The number of nutrition items that were output by the method
A	The total number of nutrition items that are in the ground truth table

ABSTRACT

Finding the right food in a supermarket for someone's dietary needs is challenging due to the large variety of food products. One possible solution is to build a nutrition information dataset and a food search engine. This engine would allow consumers to find their desired food product by placing constraints on the nutrition factors and ranking the obtained results based on their criteria. However, collecting nutrition information for tens of thousands of food products is time-consuming, thus an automatic method is desired. This work investigates the problem of automatic extraction of nutrition information from nutrition strings. For this purpose, it introduces a dataset named NutriX containing nutrition strings collected from different websites and their corresponding nutrition information. The nutrition extraction problem can be viewed as a slot filling problem, and two transformer-based methods from the literature are evaluated. This study also introduces a specialized algorithm based on dynamic programming, and evaluates it as well as the transformer-based methods and GPT-4 and GPT-4o, with encouraging results. Moreover, this study also proposes a classifier of food products that will predict the food category (e.g., fruit, pasta, ice cream, etc.) from the nutrition string. The Word2Vec, CLIP (Contrastive Language-Image Pre-Training) and DINOv3 (self-distillation with no labels version 3) language models were used in the Neural Network classifiers. Lastly, it provides a food search engine, an interactive web interface, to search for desired food products with food category and nutritional criteria. This food search engine streamlines the discovery process for consumers to identify specific food products.

CHAPTER 1

INTRODUCTION AND BACKGROUND

The United States and the entire world are facing an obesity epidemic, with more than 42% of the adults in the USA being obese (Haththotuwa, Wijeyaratne, and Senarath, 2020)(USDA, 2025). Obesity is strongly associated with heart disease, diabetes, high blood pressure, and even certain forms of cancer.

Consumers have a hard time losing weight, in part because it is hard to find what foods fit their dietary needs from the thousands of food items from supermarkets by examining their nutrition information, one item at a time. Therefore, it would be desirable that the nutrition information of all products from supermarkets be organized into a searchable database where consumers could quickly search for products satisfying their particular criteria. However, building and updating such a database requires the extraction of nutrition information from different websites containing the nutrition information of the food products, either as a text string or as a picture, or both.

1.1 Main Contribution

In this dissertation, we focus on the problem of extracting the nutrition information values of food products from text strings. Given a string containing the nutrition information (a nutrition string), the problem is to automatically extract the values of the desired fields such as calories, total fat, saturated Fat, sodium, protein, fiber, etc.

As such, this problem can be regarded as the intent classification task of a slot filling problem. In NLP (Natural Language Processing), the classification task in slot filling is to extract the values of well-defined slot types of given queries. It is referred to as the process of retrieving information in order to convert the user intent into explicit instructions. This project can be seen as a slot filling problem where the slot types are the nutrition items.

For this purpose, two popular slot filling methods are investigated: JointBERT (Chen, Zhuo, and W. Wang, 2019) and CTRAN (Rafiepour and Sartakhti, 2023), trained on our nutrition string data with slot labels. For instance, the text "Calories 90" is expected to be labeled as "Calories_item Calories_value," which offers a clear match between the nutrition item "Calories" and its value "90".

However, there are still several limitations of these slot filling labeling methods, such as increased training time for longer strings. For that reason, this dissertation also introduces in Section 3.2 a novel method based on dynamic programming to address the same problem.

This dissertation also proposes classifiers of food products that will predict the food category from the food product names using feature extractors such as Word2Vec Google news (300 features), CLIP Vit-B-32 (512 features) and ViT-L/14 (768 features) and DINOv3 ViT-L/16 (2048 features) pre-trained models. In this chapter, data augmentation and merging nutrition information with food name embedding features were implemented to mitigate the potential overfitting issues.

As the final goal, we develop of a criteria-based interaction framework that integrates an interactive frontend and a backend connected with nutrition information database to simplify searching desired food information from large-scale nutrition datasets.

In summary, this dissertation contributes the following:

- It introduces a novel knowledge extraction problem, the extraction of nutrition information values from nutrition strings, which has not received much attention in the literature.
- It introduces a dataset of 26,164 nutrition strings collected from various websites, together with their associated ground truth nutrition values.
- It shows how this problem can be viewed as a slot filling problem, evaluating state-of-the-art slot filling transformers such as JointBERT (Chen, Zhuo, and W. Wang, 2019) and CTRAN (Rafiepour and Sartakhti, 2023).
- It introduces a novel dynamic programming-based algorithm for addressing the nutrition extraction problem.
- It performs experiments on the newly introduced nutrition dataset, evaluating the proposed DP-based method, the JointBERT and CTRAN slot filling methods as well as GPT-4 and GPT-4o, which can also be used to extract information from nutrition strings.
- It also introduces the related problem of food category prediction from the food product name and experiments on the nutrition dataset, evaluating the accuracy of Neural Networks classifiers based on features extracted using the Word2Vec Google news, CLIP and DINOv3 pre-trained feature extractors.
- It constructs a guided query web interface with food categories and nutrition information constraints, effectively reducing user effort and eliminating the need for extensive manual searches to identify target food products.

1.2 Related Work

Related work can be divided into work on nutrition extraction datasets, slot filling methods and text embeddings for classifiers.

1.2.1 Nutrition extraction datasets

To our knowledge, there exists no nutrition extraction dataset similar to the one introduced in this project, containing nutrition strings and associated ground truth nutrition values. FoodDB (Harrington et al., 2019) introduces a database of UK foods and their associated nutrition information and chemical composition. The data is in the form of an online table containing the chemical compound and nutrition information of foods, without the nutrition strings. The data was collected from web pages and snapshots based on Optical Character Recognition (OCR). However, the dataset contains generic information such as the protein content in breakfast cereal and pasta rather than on specific food products.

1.2.2 Slot filling methods

There has been quite a lot of research addressing the slot filling problem. Examples include recurrent neural network (RNN) based methods such as (Liu and Lane, 2016; Mesnil et al., 2015; Y. Wang, Shen, and Jin, 2018), convolutional neural network (CNN) based methods such as (Vu, 2016; Xu and Sarikaya, 2013), and also the methods used in this dissertation, Transformer encoder-based BERT model, JointBERT (Chen, Zhuo, and W. Wang, 2019), and the Transformer-based CNN method, CTRAN (Rafiepour and Sartakhti, 2023).

JointBERT (Chen, Zhuo, and W. Wang, 2019) is a fine-tuned BERT (Devlin, M.-W. Chang, et al., 2019) model which is used to perform intent and slot classification tasks simultaneously. It performs well on widely used datasets such as ATIS and SNIPS but may be surpassed by some more recent methods.

CTRAN (Rafiepour and Sartakhti, 2023) is built on a encoder-decoder architecture with CNN and Transformers. The encoder used in this model combines the BERT embeddings with convolutional layers, and uses separate decoders for intent and slot detection. It outperformed previous slot filling methods on the ATIS and SNIPS datasets.

There are several Spoken Language Understanding (SLU) datasets widely used for evaluating slot filling methods. A related dataset called SLURP (Bastianelli et al., 2020) is a collection of audio recordings. Another dataset, SNIPS (Coucke et al., 2018), comes from a voice platform that includes queries with the associated intent outputs, such as “PlayMusics” and “GetWeather”. The ATIS dataset (Hemphill, Godfrey, and Doddington, 1990) is a series of strings for a flight-booking system, which has higher similarity outputs than the other two datasets. Both the SNIPS and ATIS datasets are widely used in Slot Filling research.

Table 1.1: A comparison of some existing Slot Filling datasets and our dataset.

Dataset	Size	Length Mean	Length Max	Slots	Slots Per String
SNIPS	14484	9.0	35	72	4.6
Atis	5871	11.1	46	120	4.1
NutriX (ours)	26164	244.0	1209	78	42.6

A data format similar to (Goo et al., 2018) for the ATIS dataset was used in this dissertation to label the nutrition extraction dataset for slot filling. One of the challenges was the length difference between the strings of these datasets and the nutrition strings. The nutrition strings are much longer (most of them over 200 words per string), and there are more slot labels in each string, which may lead to much longer computational time for training.

In Table 1.1 are shown some statistics about two existing datasets, SNIPS (Coucke et al., 2018) and ATIS (Hemphill, Godfrey, and Doddington, 1990), in comparison to the proposed NutriX dataset. In this table, the size column represents the number of strings in each dataset. The mean and maximum string length are shown in the Length Mean and Max columns respectively. Slots represent the total number of labeled categories in each dataset. Slots per String is the average number of slots in each string. From Table 1.1 one can see that the NutriX dataset is quite challenging having longer strings and more slots per string than the other two datasets.

1.2.3 Text Embeddings

There are several studies regarding words embeddings in text classification.

Bidirectional encoder representations from transformers (BERT) (Devlin, M.-W. Chang, et al., 2019) is a transformer-based language representation model, which is able to understand words based on both their meanings and the context obtained from the surrounding text. It is widely used as a pre-trained model in following NLP works.

Global vectors for word representation (GloVe) (Pennington, Socher, and Manning, 2014) is an unsupervised learning method used to generate word embeddings. It captures semantic relationships between words by utilizing the co-occurrence patterns across a large text corpus. It takes advantage of the global word statistics and leads to better word representations compared to the models relying only on the local context.

The word2vec-google-news-300 is a pre-trained Word2Vec embedding model developed by Google (Mikolov et al., 2013), which was trained on the Google News dataset with about 100 billion words. This model contains 300-dimensional vectors (features) for the specific 3 million words and phrases.

If a word falls outside the these 3 million words and phrases list, it will be impossible to look up; in these cases, people need to use alternatives models or build another custom model.

FastText (Joulin et al., 2016) is another method for embedding words developed by the Facebook AI Research lab. It breaks down a word as a bag of n-grams of characters (or sub-words), which allows it to create better vectors for rare words. However, it is usually more computationally intensive than Word2Vec, since it runs through many characters for each word.

The contrastive language-image pre-training (CLIP) model (Radford et al., 2021) is a pre-trained multi-modal neural network for matching images and text developed by OpenAI. This model includes a text encoder, a transformer model, and an image encoder, a vision transformer, which converts text and images into vectors. The models used in this dissertation are the CLIP Vit-B-32 and ViT-L/14 embedding models with 512 and 768 features, respectively.

CHAPTER 2

OVERVIEW OF THE NUTRITION EXTRACTION DATASET

The nutrition extraction dataset consists of a number of nutrition strings and their associated nutrition information values. It is organized as a table with different food products as rows, and their nutrition strings and corresponding nutrition items as columns. An example of a nutrition string is given in Figure 2.1.

Nutritional InfoIngredientsAllergensNutritional InformationServing Size: 125Calories: 90 Calories From Fat: 35Amount Per ServingPercentage Daily ValueCalcium8%Vitamin A15%Vitamin C30%Iron 4%Percentage Amount of Calories allergenno allergen information90calCalories4gTotal Fat 3g Fiber 3g Protein 410 mg Sodium

Figure 2.1: Nutrition string example.

The dataset construction can be divided into three main tasks:

1. Collection of nutrition strings from websites.
2. Semi-automatic construction of the ground truth table based on the nutrition strings.
3. Manual verification and correction of the constructed nutrition table.

These tasks will be discussed in the following subsections, together with the challenges induced by the variability of the collected nutrition strings.

2.1 Nutrition String Collection

The workflow for collecting nutrition strings from the web is illustrated in Figure 2.2.

The nutrition strings were collected from the manufacturers or retailers' websites. A web scraping technique was used to collect data from these websites with the Requests (P. S. Foundation, 2024) and Selenium (Muthukadan, 2024) Python libraries. As many websites were built using Asynchronous JavaScript and XML (AJAX) techniques, a web driver was used to control the browser in a Python environment and collect the information from the response sent by the server. Web

Figure 2.2: Workflow for collecting nutrition strings.

Table 2.1: The nutrition datasets and their sources. The length mean and max in this table are the average and max word counts for the strings in each dataset.

		length		
dataset	size	mean	max	source
Wild	2117	115	1194	Manufact
TraderJoes	1165	91	426	Retailer
Publix (Instacart)	16848	281	1209	Retailer
Target (Instacart)	6034	119	151	Retailer
Nice	24047	231	1209	Retailer
NutriX	26164	244	1209	all

pages written in AJAX can change only parts of the page without reloading the whole page as the page is scrolled down and new content is loaded from the website.

As shown in Table 2.1, four datasets were collected from different sources for a total of 26,164 strings.

The "Wild" dataset comes from more than 20 manufacturers, with 2,117 strings. Each manufacturer has a different string format, and for this reason, it is the most complex and challenging of the four datasets. The "TraderJoes" dataset with 1,166 strings, the "Publix" dataset with 16,848 strings, and the "Target" dataset were acquired from the retailers' website by web scraping. The "TraderJoes" dataset is collected from its own website (TraderJoes, 2025), while "Publix" and "Target" are from the Instacart website (Instacart, 2025). These three datasets (TraderJoes, Publix and Target) are combined into a "Nice" dataset. It was easier to extract the nutrition items and values from the strings of the "Nice" dataset than from the "Wild" dataset. The "Wild" dataset contains food product information with strings without line endings between nutrition items, while the "Nice" dataset has strings with new lines to separate the nutrition items, making it easier to label these datasets automatically based on the line-ending information.

2.2 Challenges

Extracting nutrition information from the string data faces several challenges that need to be addressed. An example of a raw nutrition string was given in Figure 2.1. In this string, words are not always separated by spaces. For instance, the word "InfoIngredientsAllergensNutritional" should be separated into 4 words. In this case, words can still be split based on the capital letters.

Generally, the nutrition values and the units of measure are placed after the nutrition item keyword in the nutrition string, but there are still some exceptions. For example, at the end of this text, the measure "90" is right before the item "Calories", and the measure "4 gram" is also located before the item "Total Fat".

Nutritional InfoIngredientsAllergensNutritionalInformation
Serving Size: 125Calories: 90 Calories From Fat: 35Calcium
8%Vitamin A15%Vitamin C30%Iron 4% information90cal
Calories4gTotal Fat 3g Fiber 3g Protein 410 mg Sodium

Figure 2.3: Nutrition string split example 1.

In Figure 2.3 is shown an example of how a string is expected to be split. Blue words are nutrition items such as Calories and Protein, red words are values, green words are the units of measure, and black words should be ignored. Each nutrition item should be matched to one value and combined with its unit, before or after the item.

Figure 2.4 is another nutrition string example. The measures are more complex than in Figure 2.3. For instance, the item "Fat" has the value "0.5g" and also "1%" after that, which means 0.5 grams is 1% of the daily value of "Fat" given by the FDA. The value "2,000" here is not supposed to match any nutrition item even though it has the nutrition item keyword "calories" after it. The values and their percentage value should also be combined together and matched to a single item.

In summary, extracting nutrition information from nutrition strings faces several challenges:

1. The name of a nutrition item in the nutrition string differs from the column name in the database. For example, the name in the database is "total fat" but the corresponding name in the nutrition string is "fat".

2. Different names of serving sizes such as pieces, tsp, and varying units of measure, e.g., mL, g, mg, oz, %, etc. Also different versions of the same unit of measure: oz, floz, fl oz, fl oz (US Ounce).

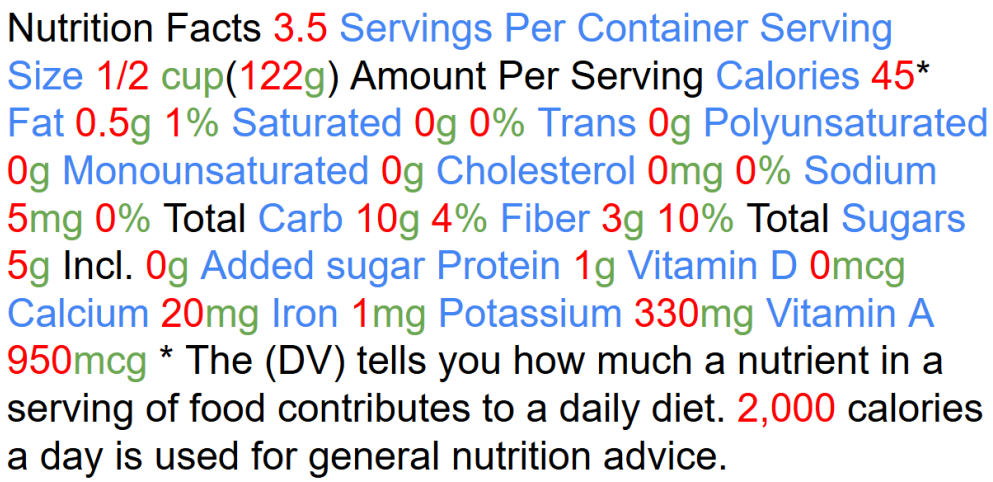

Figure 2.4: Nutrition string split example 2.

3. The placement of the associated value before or after the nutrition item name, e.g., "Calories: 90" or "Includes 0g Added Sugars".

4. No space between nutrition items and numbers, e.g., "3gProtein410mgSodium".

5. Nutrition values can be duplicated with percentage measures and should be combined together. For example, "Carbohydrates 36 g 13 %" should be combined as a single item, which means the product contains 36 grams of carbohydrates, which is 13 % of the daily value suggested by the FDA.

6. False matching of some nutrition items with others, e.g., incorrectly matching "Calories from Fat" with "Calories" and "Fat".

2.3 Constructing the Ground Truth Table

For the TraderJoes, Publix, and Target datasets, the strings were collected in two ways: one in which the nutrition items are separated by the end-of-line symbol "\n" and one as strings without end-of-line. The end-of-line format is an aid to generate the ground truth table for training and evaluation because each nutrition item and their measures with units are located on a single line. This feature is specific to the TraderJoes, Publix, and Target datasets, but not to the "Wild" dataset, and can be exploited to generate the ground truth for these three datasets. However, in Section 3.3, we will evaluate methods that are capable of processing all datasets, where each string is given as a single line.

The "Wild" dataset is collected as strings without any line breaks between nutrition items. The ground truth table for this dataset was constructed manually with the help of ChatGPT.

2.4 Dataset Verification

The verification of the correspondence between the nutrition strings and the ground truth values for the four datasets was done manually, by visual inspection of the strings and their associated values.

CHAPTER 3

NUTRITION INFORMATION EXTRACTION USING SLOT FILLING METHODS AND DYNAMIC PROGRAMMING

3.1 Slot Filling Methods for Nutrition Information Extraction

This section regards the extraction of nutrition information as a slot filling problem and shows how to train deep learning models for this purpose.

Slot filling can be seen as the labeling of an input string for subsequent extraction of the corresponding information. That is, for an input string sequence $W = (w_1, w_2, \ldots, w_n)$ with n words, the slot filling task labels each word of W with specific labels (slots), obtaining a slot sequence $S = (s_1, s_2, \ldots, s_n)$, as illustrated in Figure 3.1.

Compared to standard slot filling problems, such as ticket booking, the problem of nutrition information extraction has its own challenges because the input string is much longer (could be even 1000 words or longer) and the number of slots is quite large (78 slots).

Slot filling problems usually have two tasks: intent classification and prediction of slot labels. The intent is defined as the main purpose of the input string. In this project, the intent of a nutrition string is clear and the goal will be the prediction of slot labels. The models used in this dissertation are JointBERT (Chen, Zhuo, and W. Wang, 2019) and CTRAN (Rafiepour and Sartakhti, 2023), both having great slot label prediction performance on the Snips (Coucke et al., 2018) and ATIS (Hemphill, Godfrey, and Doddington, 1990) datasets.

3.1.1 Data Preprocessing

One major challenge for applying slot filling methods is the lack of spaces between words, nutrition items, and numbers. Capital letters are good references for inserting spaces between words. For instance, in Figure 2.1, "Nutritional InfoIngredientsAllergens" can be split into four words.

In this work, spaces are inserted before capital letters and punctuations, e.g., ":", and before and after numbers, to split the strings into words. Other methods, such as tokenizers or Bayesian

optimization (Moss et al., 2020) could be used to separate words in strings. A pretrained BERT tokenizer, such as (Moi and Patry, 2023), can also split words based on spaces and capital letters.

3.1.2 Labeling Strings for Slot Filling Training

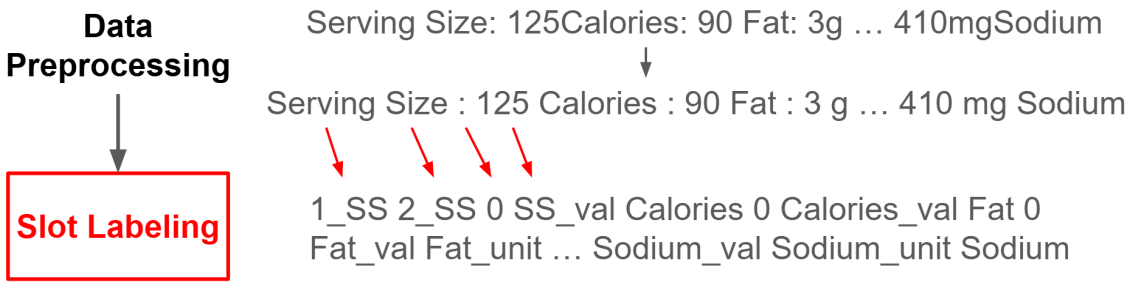

Figure 3.1: Labeling strings for training Slot Filling models.

A special type of labeling needs to be obtained for training slot filling neural networks, in which each word needs to be labeled, as illustrated in Figure 3.1 and Figure 3.2. The slot labels contain the nutrition items, values, and their units of measure. The irrelevant words are labeled with "0". The keyword "Serving Size" is a nutrition item with two words; therefore it is labeled as (1_SS) and (2_SS). In the example string in Figure 2.4, the substring "Fat 0.5g 1%" contains the keyword "Fat", measure "0.5", unit of measure "g", and the percent Daily Value (%DV) "1%", which will be defined as four slot labels, "Fat", "Fat_val", "Fat_unit", and "Fat _dv".

The slot labels for each nutrition string in TraderJoes, Publix, and Target datasets were generated from the string dataset that contained the nutrition items separated by the end-of-line symbol, as mentioned in Section 2.3.

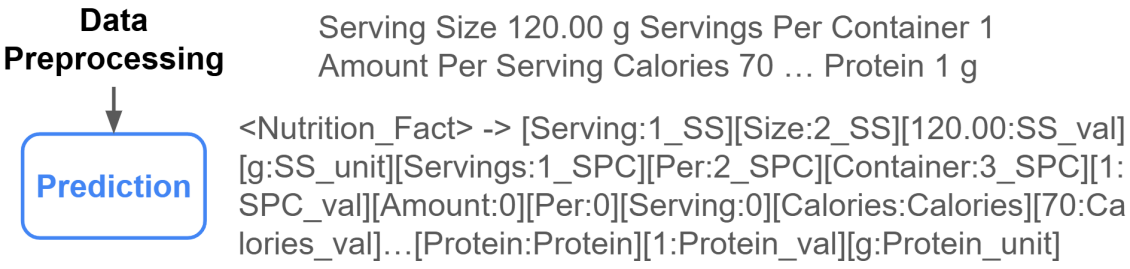

Figure 3.2: Prediction for Slot Filling models.

3.1.3 Training details

In this dissertation, the pre-trained uncased BERT model (Devlin, M. Chang, et al., 2018) was used to extract features for both JointBERT and CTRAN. The bert-base-uncased model was used for JointBERT while the bert-large-uncased model was used for CTRAN. The default training parameters were used for both methods, i.e. JointBERT was trained with initial learning rate 5e-5 using a linear schedule for 10 epochs, and CTRAN was trained with learning rate 1e-3 in encoder, 1e-4 in decoder and 1e-4 in the BERT layer using the StepLR scheduler for 50 epochs. The maximum length of the strings was set to 200, which contains almost all of the nutrition information in the nutrition datasets. Batch sizes were set to 128 for JointBERT and 16 for CTRAN.

3.2 A Dynamic Programming Method for Nutrition Extraction

This section presents an unsupervised method for slot filling based on Dynamic Programming, that does not need any training. The whole method is described in Algorithm 1. Its steps are discussed in detail in the following subsections.

Algorithm 1 Nutrition Information Extraction by Dynamic Programming.

Input: Nutrition string S

Output: Nutrition items and their values

1: Preprocess string S by adding spaces before capital letters and special characters
2: Detect nutrition values
3: Detect units of measure
4: Detect nutrition items
5: Preprocess for DP
6: Apply DP

Preprocessing uses the approach discussed in Section 3.1.1, based on adding spaces before capital letters and certain special characters.

3.2.1 Detection of Nutrition Values and Units of Measure

As Figure 3.3 shows, the first step in the DP approach is to detect the numbers and their locations in the string. The numbers are defined as contiguous sequences of characters from $\{0, 1, .., 9, \text{`.'}, \text{`-'}, \text{`/'}, \text{`\ '}, \text{`,'}\}$. These numbers include integers, decimals, numbers with thousand separators, fractions and mix-fractions such as “10”, “1.5”, “2,000”, “3/4” and “1-2/5”. They are detected together with the start and end locations in the string using regular expressions in Python. Then

the string is split into substrings based on the locations of the numbers, and hence each substring contains only words and symbols but no numbers.

Figure 3.3: Nutrition string splitting workflow.

After that, units of measure are detected whether they are right at the beginning of substrings. The units of measure are taken from a dictionary of 116 units containing “g”, “mg”, “oz”, “floz”, etc.

After the units of measure are detected, the numbers are combined with the units of measure into a single substring using “_”. For example, the value “1” combined with the unit of measure “mg” will be “1_mg”. In some cases after the numbers+unit of measure there is another number+%, the percent Daily Value (%DV), such as “0.5 g 1 %”. In such cases, all four items (two numbers and two units of measure) are combined, obtaining “0.5_g__1_%”. This whole conglomerate represents a candidate nutrition value, with its unit of measure and possibly its %DV.

3.2.2 Nutrition Item Detection

The next step is to detect whether the substrings contain any keywords for the nutrition items we are trying to extract. The nutrition items are detected using a list of 19 possible keywords, “Servings Per Container”, “Serving Size”, “Calories”, “Fat”, “Saturated Fat”, “Trans Fat”, “Cholesterol”, “Sodium”, “Carbohydrate”, “Fiber”, “Sugars”, “Added Sugars”, “Protein”, “Vitamin A”, “Vitamin C”, “Vitamin D”, “Calcium”, “Iron”, “Potassium”, and their variations, such as “fat”, “total fat”, “saturated”, “saturates”, etc. For each detected item, the start and end location in the string and the corresponding nutrition item are recorded. In the case of the ambiguous nutrition items

such as “Fat” and “Saturated Fat”, the items with more words will be detected before those with fewer words. Once any of the items is detected, the detected nutrition item will be removed from the substring and the next possible nutrition item will be searched in the current substring.

For example, in the string “Total Fat 0.5 g Total Saturated Fat 0 g Total Trans Fat 0g”, the substring “Total Saturated Fat” will be tested if “Saturated Fat” is contained. Since “Saturated Fat” is detected, it will be removed from the substring and the program will detect whether the next item ’Fat’ is contained in the substring “Total”. This method also works in the case of the substring “Servings Per Container Serving Size” in Figure 2.4, which contains more than one nutrition item. In this case, both “servings per container” and “serving size” will be detected.

The substrings that are left at the end of nutrition item detection are text to be ignored. These ignored texts will be joined with an adjacent nutrition item and will be considered as a single node for Dynamic Programming in Section 3.2.3.

3.2.3 Overview of Dynamic Programming

Dynamic programming is an efficient globally optimal algorithm for minimizing an additive cost on a chain of nodes $V_1, ..., V_n$, each having a set of possible labels (values) $l_i \in L$.

Let $\theta_k(i)$ be the unary cost for node V_k taking value i, and $\theta_k(i, j)$ be the binary cost for node V_k taking value i and node V_{k+1} taking value j.

Under these assumptions, dynamic programming obtains the global optimum of the minimization problem:

$$\min_{l \in L^n} [\sum_{i=1}^{n} \theta_i(l_i) + \sum_{i=1}^{n-1} \theta_i(l_i, l_{i+1})]. \tag{3.1}$$

It does so in a recursive manner by memorizing $M_k(j)$, the minimum cost of the partial optimization problem on $V_1, ..., V_k$ with the label of V_k being j:

$$M_k(j) = \min_{l \in L^k, l_k = j} \sum_{i=1}^{k-1} [\theta_i(l_i) + \theta_i(l_i, l_{i+1})]. \tag{3.2}$$

The vectors $M_k(j)$ are computed recursively, with the first one being $M_1(j) = 0$, and the following ones as:

$$M_{k+1}(j) = \min_{l \in L} [M_k(l) + \theta_k(l) + \theta_k(l, j)]. \tag{3.3}$$

Finally, the optimum is obtained by minimizing:

$$\min_{l \in L} [M_n(l) + \theta_n(l)], \tag{3.4}$$

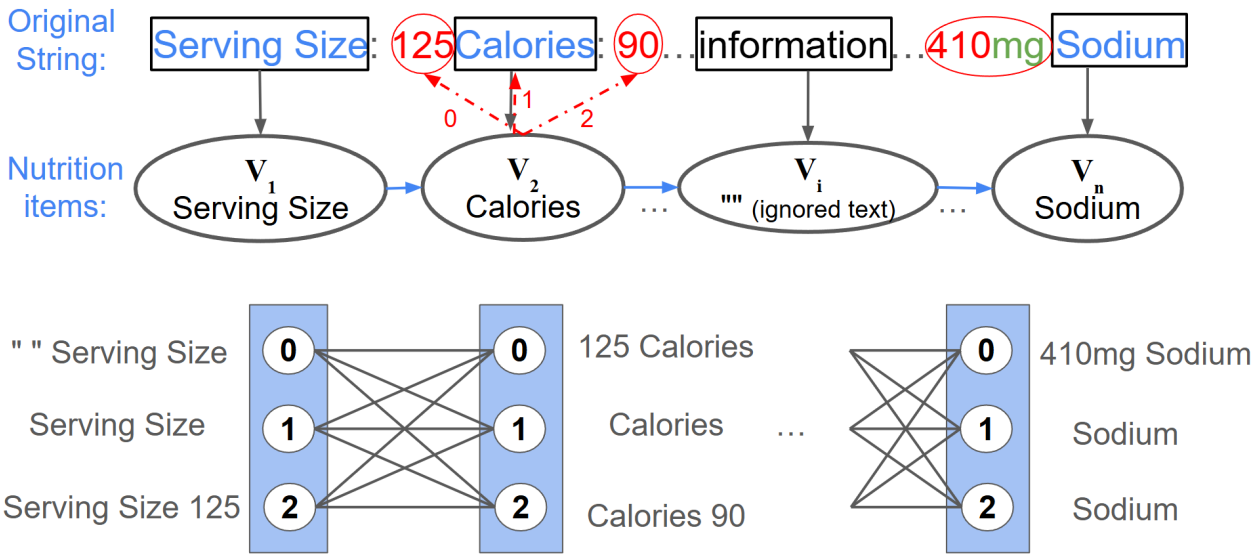

Figure 3.4: Dynamic Programming for matching nutrition item to nutrition values.

and the optimal solution can be traced back by memorizing the argmin for Eq. (3.4) and each step of Eq. (3.3), and tracing back the optimum starting from n down to 1.

3.2.4 Dynamic Programming for Matching Nutrition Items to Nutrition Values

At this step, the nutrition string has already been split into nutrition values and some nutrition items and ignored texts between them.

The problem is now how to match the nutrition items with the nutrition values, assuming that the value could be before or after the nutrition item. For that purpose, dynamic programming (DP) is used to find the matches.

As Figure 3.4 shows, to use DP to match the nutrition items to values, the DP nodes $V_1, ..., V_n$ are the detected nutrition items in the order they are in the string. The labels take three values $L = \{0, 1, 2\}$, defining whether the corresponding nutrition value is before the item ($l = 0$), after the item ($l = 2$), or it does not have a match ($l = 1$).

The unary cost $\theta_k(j)$ is the cost that a certain nutrition item matches the values before or after in the string, based on the units of measure of those values. For example, the nutrition item "Protein" is more likely to be matched to a value with unit "g" but not "mg", and hence the cost of "g" in this case will be smaller than "mg". The dimension of the unary cost $\theta_k(j)$ for each nutrition item is 3.

Table 3.1: Unary costs $\theta_k(j)$ and binary costs $\theta_k(i,j)$ used in dynamic programming.

Unit type\j	0	1	2
Preferred	1	999	1
Other	10	999	10

a) $\theta_k(j)$

$i\backslash j$	0	1	2
0	1	1	1
1	1	1	1
2	10	1	1

b) $\theta_k(i,j)$

The cost values for each nutrition item that is matched to nutrition values having different units are defined based on matching to a preferred unit of measure or not, as shown in Table 3.1, a).

The preferred units of measure are:

1. No unit for: 'Servings Per Container', 'Calories' and ignored text.
2. Any unit except no unit for 'Serving Size'.
3. 'g' for: 'Total Fat', 'Saturated Fat', 'Trans Fat', 'Carbohydrate', 'Dietary Fiber', 'Total Sugars', 'Added Sugars', and 'Protein'.
4. 'mg' for: 'Cholesterol' and 'Sodium'.
5. 'mg' or '%' for: 'Vitamin A', 'Vitamin C', 'Vitamin D', 'Calcium', 'Iron' and 'Potassium'.

The pairwise cost $\theta_k(i,j)$ is the cost of node k taking label i and node $k+1$ taking label j. If the label of V_k is 2 (i.e., "after"), there should be a high cost for label of V_{k+1} to be 0 (i.e., "before"), because that would assign the same nutrition value to two items. Thus, the cost should be larger in this case than in other combinations. The values of $\theta_k(i,j)$ in our experiments are given in Table 3.1, b).

3.3 Experiments

Experiments were performed on a Windows 11 computer with a 13th Gen Intel(R) Core(TM) i7-13700HX 2.10 GHz processor with 32GB RAM and an RTX 4060 GPU with 8GB memory.

Experiments were performed on the nutrition extraction dataset introduced in Chapter 2, with its parts shown in Table 2.1.

Because the "Wild" dataset contains food product information from strings without line breaks between nutrition items, the "Wild" dataset could not be labeled for slot filling, so it was only used as a test set for all methods.

The other three parts: TraderJoes, Publix and Target were merged into a "Nice" dataset containing 24047 observations, shown as "Nice" in Table 2.1.

The methods that were evaluated are: JointBERT (Chen, Zhuo, and W. Wang, 2019), CTRAN (Rafiepour and Sartakhti, 2023), GPT-4 (OpenAI, 2024a), GPT-4o (OpenAI, 2024b), and the proposed dynamic programming (DP) method described in Section 3.2.

GPT-4 and GPT-4o were queried using Python and their respective API from the OpenAI library (`https://openai.com/api/`). Each prompt in the GPT-4 and GPT-4o API includes the model to be used, i.e., "gpt-4", "gpt-4o" and "gpt-4o-mini", and the message part with the nutrition strings and the request to extract the nutrition items and their values. The response of the prompts were collected and evaluated using the GT table.

The other three methods were implemented in Python. For JointBERT and CTRAN, their GitHub implementations from `https://github.com/monologg/JointBERT` and `https://github.com/rafiepour/CTRAN` were used.

JointBERT and CTRAN were trained as described in Section 3.1. Training each fold of the JointBERT took an average of 1.1 hours for 10 epochs and for CTRAN took 16.5 hours for 50 epochs.

The methods that require training (JointBERT and CTRAN) were evaluated with four-fold cross-validation on the "Nice" dataset. The models trained on each of the folds were also used to evaluate the "Wild" dataset.

The unsupervised methods (GPT-4, GPT-4o and DP) were evaluated on the entire dataset.

The DP algorithm took around 49 msec per query while the JointBERT took 13 msec and CTRAN took 396 msec per query. Querying GPT-4 using the API took about 3724 msec while GPT-4o took about 4097 msec per query.

3.3.1 Evaluation Measure

The ground truth (GT) table of each dataset was used to create the baseline for evaluation and to calculate the precision and recall of each food product (row) by matching the predicted and GT values for each nutrition item. The overall precision and recall are computed by averaging the row-wise precision and recall values, and the F_1-score is calculated based on the overall precision and recall.

The precision of a method is defined as the ratio $p = M/N$, where N is the number of nutrition items that were output by the method, and M is the number of correct nutrition items that were output by the method according to the GT table.

Table 3.2: Nutrition extraction evaluation. Shown are the precision, recall, and F_1-scores of the five methods, with best results in bold, the second best results in blue text.

Method	Precision (std.)	Recall (std.)	F_1 (std.)
Nice dataset, N=24047			
DP	98.9	95.4	97.1
JointBERT	98.6(0.01)	89.9(0.05)	94.0(0.02)
CTRAN	**99.9(0.01)**	**99.9(0.01)**	**99.9(0.00)**
GPT-4	95.5	83.1	88.9
GPT-4o	95.4	90.0	92.6
Wild dataset, N=2117			
DP	**97.5**	**98.6**	**98.1**
JointBERT	76.6(0.02)	70.5(0.03)	73.4(0.02)
CTRAN	86.8(0.03)	79.5(0.04)	83.0(0.03)
GPT-4	90.3	80.9	85.4
GPT-4o	92.2	85.7	88.8

The recall is defined as the ratio $r = M/A$, where A is the total number of nutrition items that are in the GT table, and M is the number of items from the GT table that are found in the output.

The F_1 measure has the usual definition in terms of the precision p and recall r, $F_1 = 2pr/(p+r)$.

3.3.2 Results

Experimental results were evaluated for the values of the following nutrition items: Servings Per Container, Serving Size, Calories, Fat, Saturated Fat, Trans Fat, Cholesterol, Sodium, Carbohydrate, Dietary Fiber, Sugars and Protein.

The results are shown in Table 3.2, with the methods that require training having mean (and std.) test values on the four cross-validation folds.

From Table 3.2 one can see that the F_1 scores on the "Nice" dataset are almost consistently better than those on the "Wild" dataset because the "Nice" dataset strings are more homogeneous, hence easier to process. GPT and GPT-4o have good performance on the "Nice" dataset, and both JointBERT and CTRAN have similar precision results, while CTRAN has better recall on both the "Nice" dataset and the "Wild" dataset. The proposed DP method has a stable performance on both datasets, with the second best results for the 'Nice' dataset, and outperforms the other methods on the more challenging "Wild" dataset. For CTRAN, 50 epochs were enough because the cross-validated results on the "Nice" dataset are extremely good. However, the generalization of CTRAN to the "Wild" dataset is not as good as of the DP method, which is an unsupervised method with very few tuning parameters.

Table 3.3: Ablation study on the Wild dataset.

Method	Prec	Recall	F_1
DP	97.5	98.6	98.1
No preprocessing	95.2	93.1	94.1
No binary cost	76.8	72.9	74.8

GPT-4 and GPT-4o provide a convenient Python API environment for sending input prompts containing the nutrition strings and the request. But one disadvantage of GPT-4 is that each prompt is irrelevant to others, which means that the user has to give very specific prompts such as: "Extract the nutrition item Protein from this string." for each string; otherwise one cannot expect stable results. Moreover, although it is not necessary to train a model on GPT-4 and GPT-4o, it takes a much longer time querying strings than the other methods. ChatGPT does have a good performance on extracting the nutrition items from strings, but users have to type in the strings repeatedly in the message bar manually unless using a web driver.

Ablation study. An ablation study of the DP method introduced in this dissertation is shown in Table 3.3. It evaluates the importance of the preprocessing step and the binary cost in the obtained result. The performance is poor without the binary cost from Table 3.1. In this case, nutrition items may be matched to same values in the strings. The preprocessing step is less important than the binary cost. However, this step was also necessary for slot filling methods, i.e., JointBERT and CTRAN.

CHAPTER 4

CLASSIFICATION OF THE FOOD PRODUCT CATEGORIES

This chapter introduces a classifier of food products that will predict the food category (e.g., pizza, hot dogs, ice cream, etc.) from the food product names. The Word2Vec (Mikolov et al., 2013), CLIP (Contrastive Language-Image Pre-Training) (Radford et al., 2021) and DINOv3 (Siméoni et al., 2025) language models will be used as word embeddings in these classifiers.

4.1 The Master List of the Food Categories

As the food product names were collected from separate websites, different categories were defined by different websites for identical food products. For example, a chocolate chip cookie can be classified as a "cookie" on some websites but may also be classified as a "cookie and candy" on other websites. Therefore, a master list of category names is necessary for the whole dataset to train and evaluate the classifiers. This list is defined based on the union of food categories on "Publix", "Target" and "Trader Joe's" websites. It contains 100 categories of food products. The detailed list is given in Appendix A.1.

4.2 The Word Embedding Methods

Words with similar meanings and frequencies tend to be grouped together. To quantify further research and perform statistical analysis with these groups, there is interest to assign a numeric vector to each word, and this numeric vector is called word embedding. Under this embedding, similar words should be grouped nearby based on the the distance in the embedding space. With the relationship, researchers can perform language research such as text classification and other applications.

Word embeddings primarily quantify the information of a word's context based on its usage in the text, which allows subsequent advanced NLP applications to be performed using this information. In this section, Word2Vec, CLIP and DINOv3 models are introduced and applied to the food product name classifiers.

4.2.1 Word2Vec

Word2Vec (Mikolov et al., 2013) is a type of language model that learns semantic knowledge from large amounts of text in an unsupervised manner, which is widely used in natural language processing. It is used to generate word embeddings, which are closely related to language models. This method converts words into vectors, which can simplify the processing of text content into vector operations in the vector space, calculate the similarity in the vector space, and express the semantic similarity of the text.

In this dissertation, we will use the Word2Vec function from the Gensim(Řehůřek and Sojka, 2010) package, where Gensim is a useful Python NLP package that encapsulates Google's word2vec version, and there are many other useful word vector APIs such as Doc2Vec, FastText that are available in this package.

4.2.2 CLIP

CLIP (Contrastive Language-Image Pre-Training) model (Radford et al., 2021), which was released by OpenAI in early 2021, is a pre-trained multimodal neural network model for matching images and text. This model directly uses a large amount of internet data for pre-training and has achieved state-of-the-art performance on many tasks.

The text encoder in the CLIP model with Vit-B-32 (512 features) and ViT-L/14 (768 features) will be used for word embeddings in this project.

4.2.3 DINOv3

DINOv3 (Siméoni et al., 2025), released by Meta AI in August 2025, is a massive, self-learning AI model that understands images without the need for human labels. It scales up to 7 billion parameters and introduces Gram Anchoring to ensure crisp detail even in ultra-high-definition data. Because it acts as an all-in-one "foundation", a single version of the model can handle anything from object detection to depth mapping, removing the need for expensive, task-specific fine-tuning.

However, DINOv3 does not provide a built-in language interface for text. Thus, in this study, dino.txt (Jose et al., 2024), which is a framework designed to give a language interface to DINOv2 (Oquab et al., 2024), the previous version of the DINOv3 vision model, will be used to create text embeddings for DINOv3. Although the original DINOv2 only recognized images, dino.txt connects the gap between pure vision and language by aligning a text encoder with the existing DINOv2 architecture. By utilizing a frozen visual backbone and a specialized "[CLS avg]" concatenation

pooling method, the framework enables both global and pixel-level understanding without extensive retraining. This approach is highly efficient—matching CLIP's capabilities and supporting open vocabulary applications, allowing precise zero-shot classification and dense segmentation based on arbitrary text prompts.

4.3 Data Augmentation

Figure 4.1 and Figure 4.2 show the loss and accuracy curves of the neural networks with 2 hidden layers between training and validation with a Linear Scheduler and start factor = 1, end factor = 0.01 and learning rate = 0.0001 in 300 epochs using the DINOv3 2048 features pre-trained model in blue solid lines. The gaps between training and validation in both loss (0.0145 for training and 0.8170 for validation) and accuracy (99.18 for training and 88.82 for validation) are evidence that overfitting issues may exist during training.

To prevent overfitting and improve generalization, this dissertation integrated EDA (Easy Data Augmentation) (Wei and Zou, 2019) and TextAugment (Marivate and Sefara, 2020) strategies into the training pipeline of neural networks to synthetically expand the dataset. These methods force the model to prioritize semantic meaning over specific syntax. Synonym replacement, random swap, and random insertion strategies in EDA will be used to introduce lexical and structural variety while preserving the original sentiment of the text. The punctuation insertion strategy in TextAugment will also be used to simulate real-world noise, serving as a regularization tool. Data augmentation will increase the diversity of the training distribution, narrowing the gap between training and validation accuracy to ensure resilience against unseen linguistic variations.

To enhance the robustness of the neural network, this dissertation employs a dynamic data augmentation framework. During the training phase, each product name is subjected to one of four data augmentation strategies (synonym replacement, random swapping, random insertion, and punctuation insertion) or retained in its original form. By applying these techniques stochastically, the model encounters a different variation of the same data point across separate training epochs. For example, the product "Heinz Tomato Ketchup" might be transformed into "Heinz Tomato? Ketchup" via punctuation insertion in the first epoch, while the second epoch might present it as "Heinz Ketchup Tomato" through a random swap strategy. This approach prevents overfitting and encourages the model to learn noise-invariant semantic features.

The green dotted lines in and Figure 4.2 shows the accuracy curves of the neural networks with 2 hidden layers between training and validation with Linear Scheduler and start factor = 1, end

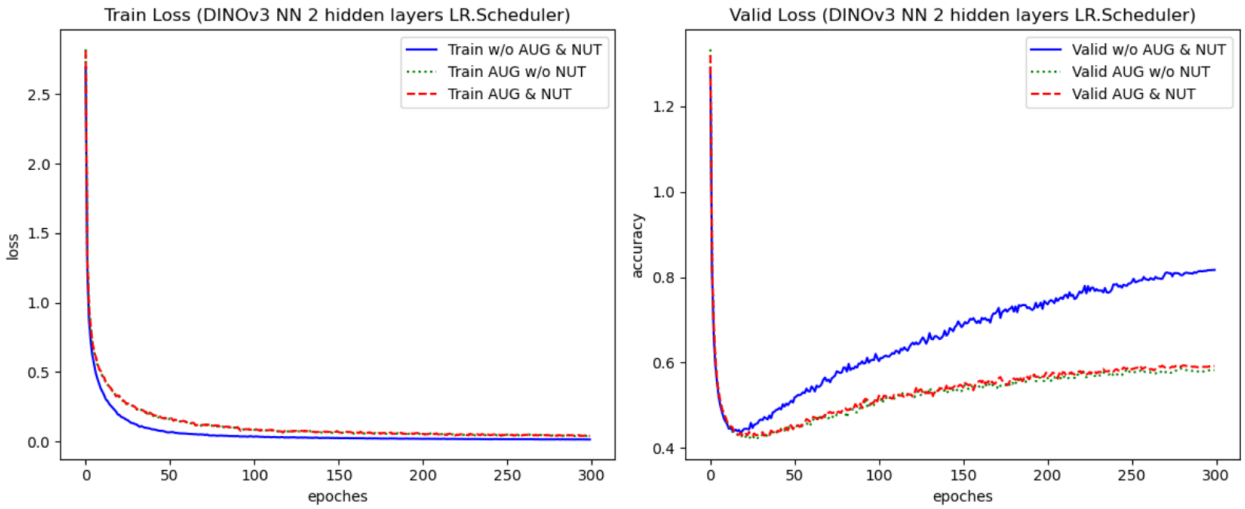

Figure 4.1: DINOv3: 2 hidden layers Neural Network loss curve. Comparing with or without data augmentation and nutrition information with Linear Scheduler and learning rate = 0.0001 in 300 epochs.

factor = 0.01 and learning rate = 0.0001 in 300 epochs using the DINOv3 2048 features pre-trained model with data augmentation. The validation curves has improved compared to the experiment without data augmentation in blue solid line.

4.4 Including nutrition information as features

Nutrition data provides an objective framework for differentiating food groups. A clear illustration is the contrast between carbonated soft drinks and juices: while both may contain sugar, soft drinks often provide more Calories and Sugars than juice. In this dissertation, standard word embeddings are augmented with these specific nutritional attributes sourced from the Nutrix dataset in Chapter 3. By combining semantic textual data with these quantitative nutrition information for food products, the classifier can distinguish between similar but nutritionally distinct products more effectively.

Similarly Section 4.3, the red dashed lines in Figure 4.2 show the accuracy curves of the neural networks with 2 hidden layers between training and validation with using the DINOv3 2048 features pre-trained model with data augmentation and nutrition information. The validation accuracy has improved even further compared to the experiment in Section 4.3.

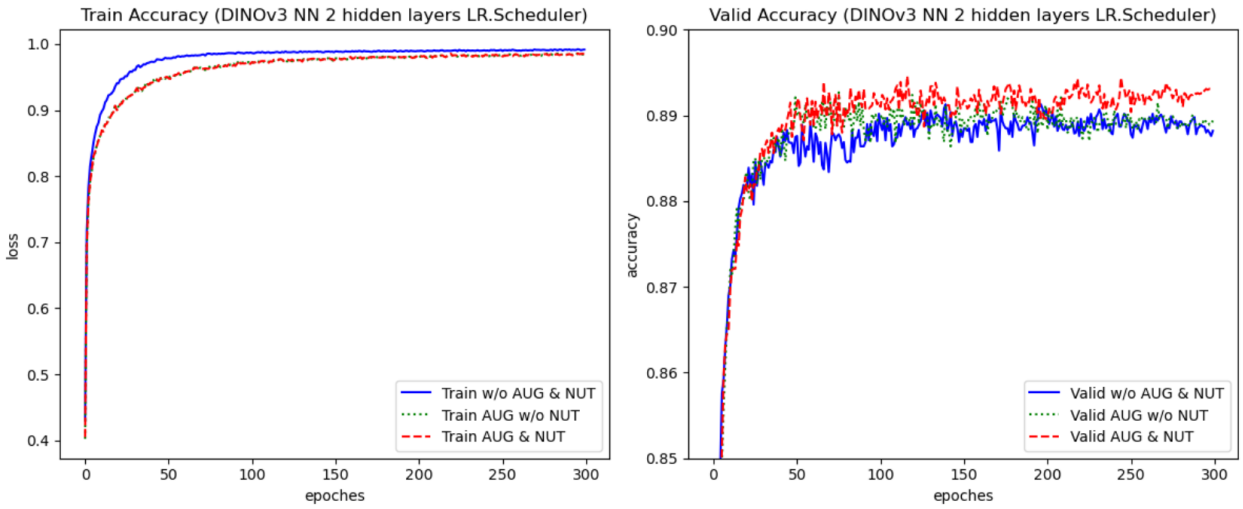

Figure 4.2: DINOv3: 2 hidden layers Neural Network accuracy curve. Comparing with or without data augmentation and nutrition information with Linear Scheduler and learning rate = 0.0001 in 300 epochs.

4.5 Training Details of the Classifiers

For the training process, 70% of the 26,164 food products are used as the training set while 30% are used as the validation set. The batch size of the Neural Network is 100. We experiment a one hidden layer (a 512 neuron fully connected layer followed by ReLU (Rectified Linear Unit) activation function and another fully connected layer), two hidden layers (with hidden size 512 and 256 both followed by ReLU activation function) and three hidden layers (with hidden size 512, 256 and 128 all followed by ReLU activation function) Neural Networks. The original learning rate is 0.0001 using linear scheduler with start factor 1 and end factor 0.01 in 300 epochs. The training time for Neural Networks is presented in Table 4.1.

Table 4.1: Training time (seconds) for Neural Networks with learning rate 0.0001 of the 1 to 3 hidden layer Neural Networks.

Method	2-layers	3-layers	4-layers
Neural Networks without Data Augmentation and Nutrition Information			
W2v	249	414	480
CLIP512	267	395	466
CLIP768	266	373	411
DINOv3	804	1,346	1,258
Neural Networks with Data Augmentation			
W2v	3,525	3,789	3,856
CLIP512	3,714	3,859	4,028
CLIP768	6,127	6,481	6,543
DINOv3	56,970	57,236	57,359
Neural Networks with Data Augmentation and Nutrition Information			
W2v	3,364	3,417	3,485
CLIP512	3,754	3,806	3,997
CLIP768	6,218	6,680	6,862
DINOv3	56,805	57,287	57,265

4.6 Results

This section compares the performance of 1 to 3 hidden layer neural networks using an initial learning rate of 0.0001 with Linear scheduler. A comprehensive table including the classification results for neural networks under different learning rates and schedulers is presented in Appendices A.2, A.3, A.4.

Table 4.2: Food category classification evaluation. Shown are the training and validation accuracy of the 1 to 3 hidden layer Neural Networks (Linear Scheduler with start factor = 1, end factor = 0.01 and initial learning rate = 0.0001 in 300 epochs) method, with best results in bold.

Method	Training Accuracy	Validation Accuracy
Word2Vec (Gensim) with Google news 300 features		
2-layer NN	88.32	80.99
3-layer NN	83.39	79.77
4-layer NN	82.84	78.54
CLIP Vit-B-32 with 512 features		
2-layer NN	98.91	85.80
3-layer NN	96.18	86.65
4-layer NN	97.59	85.44
CLIP ViT-L/14 with 768 features		
2-layer NN	99.12	85.80
3-layer NN	98.36	86.55
4-layer NN	87.03	86.08
DINOv3 ViT-L/16 with 2048 features		
2-layer NN	99.03	88.37
3-layer NN	**99.18**	**88.82**
4-layer NN	99.11	88.71

The results of the classifications of food product names into 100 categories are shown in Table 4.2, with the methods Word2Vec with Google news 300 features, CLIP Vit-B-32 model with 512 features and CLIP ViT-L/14 model with 768 features. In this table, the training and validation accuracy for the 2 hidden layer neural networks using the DINOv3 2048 features pre-trained model is better than that using Word2Vec (Gensim) with Google news 300 features and CLIP Vit-B-32 and ViT-L/14 model. DINOv3 model has the best performance in both training and validation accuracy with the 2 hidden layer neural networks. The DINOv3 model has stable performance in training accuracy and the validation accuracy is as high as 88.82%, which is the best validation accuracy with neural networks compared to other models and classifiers.

However, in these neural network classifiers, the differences between training and validation accuracy are quite large as Figure 4.2 shown, thus there may still be overfitting issues to be solved.

Table 4.3: Food category classification evaluation. Shown are the training and validation accuracy of the 1 to 3 hidden layer Neural Networks (Linear Scheduler with start factor = 1, end factor = 0.01 and origin learning rate = 0.0001 in 300 epochs) method with Data Augmentation, with best results in bold.

Method	Training Accuracy	Validation Accuracy
Word2Vec (Gensim) with Google news 300 features		
2-layer NN	73.64	72.37
3-layer NN	67.53	69.89
4-layer NN	64.61	66.90
CLIP Vit-B-32 with 512 features		
2-layer NN	94.99	86.41
3-layer NN	92.36	86.75
4-layer NN	93.37	86.08
CLIP ViT-L/14 with 768 features		
2-layer NN	97.27	86.90
3-layer NN	95.57	87.87
4-layer NN	96.24	87.40
DINOv3 ViT-L/16 with 2048 features		
2-layer NN	**99.01**	88.54
3-layer NN	98.54	88.94
4-layer NN	98.48	**89.12**

The results of the classifications of the neural networks with data augmentation are shown in Table 4.3. In this table, the training and validation accuracy for all of the neural networks using the DINOv3 2048 features pre-trained model is better than that using Word2Vec (Gensim) with Google news 300 features and CLIP Vit-B-32 and ViT-L/14 model. The validation accuracy of DINOv3 model with 3 hidden layer neural network reached 89.12%, an improvement over 88.71% in Table 4.2. Furthermore, excluding the Word2Vec Google news 300 features model, a consistent improvement in validation accuracy was observed in the rest of the experimental group, implying a reduction in overfitting.

Table 4.4: Food category classification evaluation. Shown are the training and validation accuracy of the 1 to 3 hidden layer Neural Networks (Linear Scheduler with start factor = 1, end factor = 0.01 and origin learning rate = 0.0001 in 300 epochs) method with Data Augmentation and Nutrition Information, with best results in bold.

Method	Training Accuracy	Validation Accuracy
Word2Vec (Gensim) with Google news 300 features		
2-layer NN	75.62	75.16
3-layer NN	70.67	73.52
4-layer NN	65.11	69.95
CLIP Vit-B-32 with 512 features		
2-layer NN	95.45	86.61
3-layer NN	92.58	87.26
4-layer NN	93.08	86.93
CLIP ViT-L/14 with 768 features		
2-layer NN	97.39	87.01
3-layer NN	95.56	87.71
4-layer NN	96.44	87.73
DINOv3 ViT-L/16 with 2048 features		
2-layer NN	**99.02**	88.68
3-layer NN	98.52	**89.31**
4-layer NN	98.59	88.66

The results of the classifications of the neural networks with data augmentation and nutrition information are shown in Table 4.4. In this table, the training and validation accuracy for all of the neural networks using the DINOv3 2048 features pre-trained model is also better than that using other models. The validation accuracy of DINOv3 model with 2 hidden layer neural network reached 89.31%, which is a noticeable improvement compared to 88.82% in Table 4.2 and 88.94% in Table 4.3. All validation accuracy except the 2 hidden layer neural networks using CLIP ViT-L/14 model performed better than that in Table 4.3, suggesting further mitigation of the overfitting issue.

CHAPTER 5

THE FOOD SEARCH ENGINE

For the ultimate goal, we created a web-based nutrition search tool that helps users find food products by applying filters such as calories, fat, etc. within selected categories. To enhance the accessibility and efficiency of large-scale nutrition data retrieval, an interactive, user-guided filtering framework is developed to replace traditional table-based browsing. This framework optimizes the query process by utilizing a sequence of constrained inputs, eliminating the need for manual scanning from large-scale nutrition datasets.

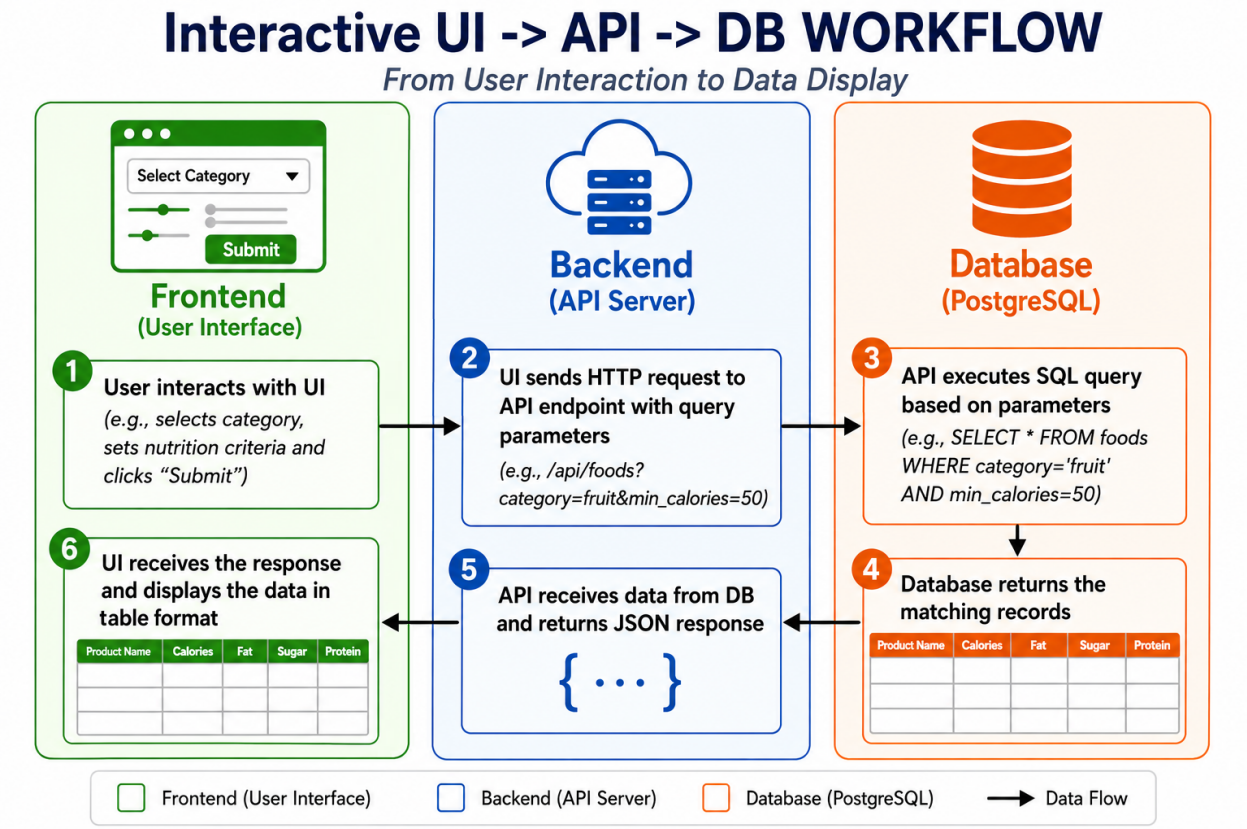

Figure 5.1: The user request workflow of the Nutrix food search engine. It shows the interaction between the frontend interface, backend API server, and PostgreSQL database during nutrition-based food filtering from users.

As shown in Figure 5.1, the workflow begins when a user selects a food category and enters

nutrition constraints through the frontend interface. JavaScript processes the inputs and sends an HTTP request to the backend API hosted on cloud platform. The backend dynamically constructs SQL queries and retrieves matching records from the PostgreSQL database. The query results are then returned as JSON responses and displayed in the frontend table interface built by a custom vanilla JavaScript table renderer.

5.1 Database

To achieve this goal, the first task is to set up a secure database on the cloud, that is, creating a cloud-hosted food product database that is not directly accessible from the public internet. A cloud provider, such as Google Cloud (Google, 2026), Microsoft Azure (Microsoft, 2026) or Amazon Web Services (Amazon, 2026), are popular database choices. These managed database services handle security, backups, and scaling for their users. In these database services, a database instance will be created with database engine such as MySQL (Oracle, 2026) and PostgreSQL (Group, 2026) under the cloud provider's environment. In this dissertation, PostgreSQL in the Render platform (Render, 2026) serves as the database engine storing nutrition information tables alongside the minimum and maximum criteria for each category. The foods table (Table 5.1) contains the whole Nutrix dataset with 26,164 food products and 100 food categories.

Table 5.1: The list of foods table in the PostgreSQL database.

Foods

Fields	**Type**	**Notes**
product_name	text	Food product names
categories	text	100 categories in total
calories	double precision	
fat	double precision	Default unit g
saturated_fat	double precision	Default unit g
trans_fat	double precision	Default unit g
cholesterol	double precision	Default unit mg
sodium	double precision	Default unit mg
carbohydrate	double precision	Default unit g
fiber	double precision	Default unit g
sugars	double precision	Default unit g
protein	double precision	Default unit g
data_source	text	Source of the food product

The table of criteria (Table 5.2) enables the backend to load the minimum and maximum value of nutrition items for each category efficiently, which helps users setting nutrition information ranges to filter foods table (Table 5.1) in the frontend.

Table 5.2: The list of criteria table in the PostgreSQL database.

Criteria

Fields	Type	Notes
categories	text	100 categories in total
min_calories	double precision	
max_calories	double precision	
min_fat	double precision	
max_fat	double precision	
min_sfat	double precision	Minimum value of Saturated Fat
max_sfat	double precision	Maximum value of Saturated Fat
min_tfat	double precision	Minimum value of Trans Fat
max_tfat	double precision	Maximum value of Trans Fat
min_chole	double precision	Minimum value of Cholesterol
max_chole	double precision	Minimum value of Cholesterol
min_sod	double precision	Minimum value of Sodium
max_sod	double precision	Minimum value of Sodium
min_carb	double precision	Minimum value of Carbohydrate
max_carb	double precision	Minimum value of Carbohydrate
min_fiber	double precision	
max_fiber	double precision	
min_sug	double precision	Minimum value of Sugars
max_sug	double precision	Minimum value of Sugars
min_pro	double precision	Minimum value of Protein
max_pro	double precision	Minimum value of Protein

5.2 Backend

The backend language framework such as JavaScript or Python is used to build endpoints that the front-end can send requests to. The connection is established to the database via the API under the private network credentials provided by the managed database services.

In this dissertation, the backend component is implemented as an API server using Node.js (O. Foundation, 2026b) and Express.js (O. Foundation, 2026a) to provide scalable and efficient request handling. Node.js is an open-source, cross-platform runtime that lets people run JavaScript on servers instead of just in browsers. It is built for creating high-performance, scalable network apps, allowing developers to use one language for both the front and back end. Express.js is a lightweight

and flexible web framework for Node.js known for its speed. It serves as the primary standard for developers building APIs and server-side applications within the JavaScript environment.

This backend function serves as an intermediary between the user-facing interface and the underlying database system. While receiving requests from the client application, the API server parses user-defined query parameters and processes the corresponding filtering logic. These filters are translated into database queries to retrieve records that satisfy the specified constraints.

Once the requested data are obtained, the server formats the results into a structured response, typically in JSON format, and transmits them back to the client for presentation. This framework separates application logic from the frontend interface, while also enabling scalability to handle large-scale nutrition datasets and multiple concurrent user requests.

5.3 Frontend

The backend API is built to act as a secure middleman that receives requests from the website and queries the database, while the frontend component is implemented as a web-based user interface that reacts to user interaction with the system. In this dissertation, the frontend interface is implemented using HTML5, CSS3, and vanilla JavaScript. Dynamic data retrieval is performed using the Fetch API, while JSON responses from the backend are rendered into HTML tables through DOM manipulation techniques. There is no external frontend table-rendering frameworks used in this interface, which allows flexible client-side processing. This frontend API includes structured input controls, including drop-down menus for category selection, input fields for specific numerical criteria for nutrition items, and a dynamic table display for presenting query results.

Users can begin the process by selecting a preferred food category, which narrows the dataset to a specific, relevant subset. Based on this selection, the system dynamically presents valid value ranges for nutrition items such as calories, sugars, and protein. Afterwards, users can define specific criteria by setting ranges or thresholds for their chosen nutrition items. During the submission process, the system filters the data and retrieves a subset of food items that meet these constraints. To support efficient comparison and decision-making for the users, these results are displayed in a structured tabular format.

5.4 Example of the Food Search Engine

The interactive Food Search Website is available on GitHub at the following URL: `https://github.com/ptlu-exp/Food-Search-Website`. To demonstrate the application of the food search framework, this section provides an example on the platform, which assists users in discovering desired food products by combining categorical filters with specific nutritional criteria. The user interaction begins with an initial dataset reduction to isolate relevant items. As a first step, users can select a preferred food category on the food search engine website. In this case, the user selects the "Ice Cream" category, which contains 726 products.

As shown in Figure 5.2, the system automatically presents the maximum and minimum values for each nutritional item within the ice cream category. This immediate visual feedback helps users understand the nutritional limits of their selected category, enabling them to set realistic criteria to find their ideal food product.

To execute a highly targeted search, the user then applies specific nutritional constraints to the active subset. In this example, the calorie limit was restricted to a maximum of 100 calories. Following this step, the food search engine reduced the pool of available choices to a final output of 114 eligible ice cream products. To demonstrate the output formatting of the platform, a results table displaying 10 of these 114 filtered products is provided in Figure 5.3.

The resulting table interface allows users to set sorting preferences for each column, including alphabetical and numerical ordering.

In conclusion, this example demonstrates how the proposed food search framework enables users to efficiently observe relevant products through category selection and nutritional filtering. By guiding users with nutritional criteria and interactive filtering tools, the food search engine platform improves efficiency of decision-making in food product selection.

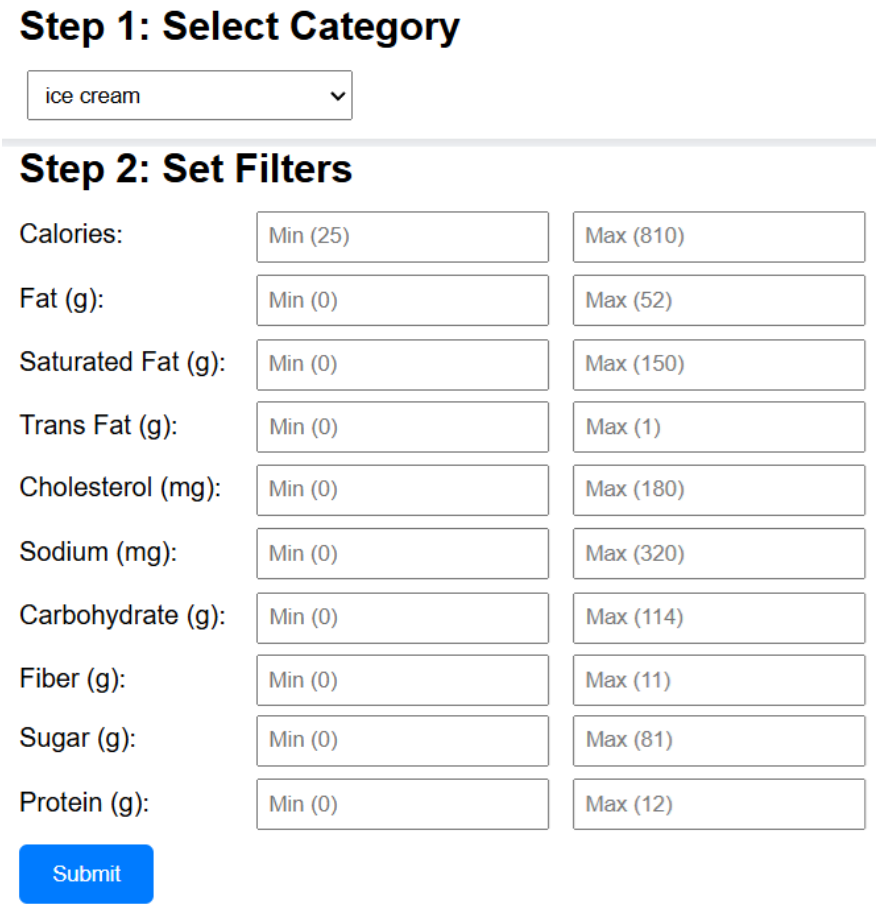

Figure 5.2: Example of the user interface. Step 1 allows user to select a food category and the maximum and minimum of each nutrition item will present in step 2.

Results

Product Name	Categories	Calories	Fat (g)	Saturated Fat (g)	Trans Fat (g)	Cholesterol (mg)	Sodium (mg)	Carbohydrate (g)	Fiber (g)	Sugars (g)	Protein (g)	Data Source
Alden's Organic Vanilla Mini Squares	ice cream	100	4.5	2.5	0	20	55	15	0	9	2	Publix
Bomb Pop Original Ice Pops	ice cream	40	0	0	0	0	15	29	0	21	0	Target
Bomb Pop Original Ice Pops	ice cream	40	0	0	0	0	15	29	0	21	0	Publix
Breyers Frozen Dairy Dessert, Vanilla, Chocolate, Strawberry	ice cream	100	6	4	0	10	40	16	4	4	2	Target
Breyers Frozen Dairy Dessert, Vanilla, Chocolate, Strawberry	ice cream	100	6	4	0	10	40	16	4	4	2	Publix
Breyers Ice Cream Natural Vanilla Snack Cups	ice cream	100	5	3.5	0	20	30	11	0	10	2	Publix
Budget Saver Monster Pops Slushed Cherry-Pineapple	ice cream	60	0	0	0	0	5	16	0	16	0	Publix
Caramel Apple Mochi	ice cream	80	2	1.5	0	5	15	14	0	10	1	Trader Joe's
Chloe's Mango Pops	ice cream	60	0	0	null	0	0	16	1	13	0	Publix
Chloe's Pops, Dark Chocolate	ice cream	60	0	0	0	0	0	15	1	13	1	Publix

Figure 5.3: Example of the resulting table. Ten food products are shown with the ice cream category and maximum calories 100.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

This dissertation introduced the NLP problem of extracting nutrition information from nutrition strings, which is a slot-filling problem that has not received attention in the NLP literature. In this regard, it provides a nutrition extraction dataset called NutriX containing more than 26,000 nutrition strings and their associated ground truth nutritional information. It shows how the dataset was collected and annotated, and the challenges facing this problem compared to other slot filling problems, challenges such as long strings and a large number of slots per string.

It also shows how to view the nutrition extraction problem as a slot filling problem and how to train deep learning models on this data. Moreover, this dissertation introduces a novel method based on dynamic programming for the same task.

Experiments on the provided dataset show that the problem is not trivial, and one part of the dataset called the "Wild" dataset is quite challenging where the proposed DP method works best among the five methods evaluated, including two GPT-4 versions.

This dissertation also studied the problem of food category classification using Neural Network classifiers. The food products were labeled with a predefined master list of food categories among the three main web sources Publix, Target and Trader Joe's. The Word2Vec Google News model with 300 features, CLIP Vit-B-32 with 512 features, ViT-L/14 with 768 features and DINOv3 ViT-L/16 with 2048 features were used as word embeddings in the experiments. The DINOv3 ViT-L/16 model has stable and good performance in training accuracy and the highest validation accuracy with neural networks compared to other models and classifiers. However, the significant disparity between training and validation accuracy in these neural network classifiers suggests the presence of overfitting, which necessitates further resolution. Therefore, this dissertation utilizes a dynamic data augmentation framework and integrates nutritional data as input features. The data augmentation strategy enriches the training distribution diversity, reducing the performance gap between training and validation sets to improve resilience against novel linguistic variations. Furthermore, by merging semantic text with quantitative nutrition information, the classifier achieves

higher validation accuracy comparing to experiments without data augmentation and nutrition information.

Finally, this dissertation implemented a framework integrating a web-based frontend, a backend API server built with Node.js and Express.js, and a structured data layer to support efficient querying of large-scale nutrition datasets. The frontend provides an straightforward interface through drop-down food category selection, nutrition criteria input fields and dynamic tables, which enables users to specify category-based and nutritional constraints. The backend processes these inputs, applies filtering logic, and retrieves relevant records from the database. This criteria-based interaction framework reduces user effort, supports guided interaction, and retrieves relevant information from large-scale nutrition datasets more effectively, and helps people finding out desired food product without wasting time on extensive searches for desired food products.

APPENDIX A

APPENDIX

A.1 Master List of Food Product Categories

'asian sauce', 'baking powder', 'bar', 'barbecue sauce', 'bean', 'beef and lamb', 'beer', 'bouillon', 'bread', 'bread crumb', 'butter', 'cake', 'candy', 'canned vegetable', 'cereal', 'cheese', 'chicken', 'children health', 'chip', 'chocolate', 'coffee', 'coffee creamer', 'cone', 'cookie', 'cracker', 'cream', 'deli meat', 'dip sauce', 'dough', 'drink mix', 'egg', 'energy drink', 'fish', 'flatbread', 'flour', 'frosting', 'frozen vegetable', 'fruit', 'fruit snack', 'gum and mint', 'half & half', 'hard beverage', 'health care', 'honey', 'hot dog', 'hot sauce', 'ice cream', 'instant food', 'jam', 'jelly', 'juice', 'ketchup', 'milk', 'mustard', 'non-alcoholic drink', 'noodle', 'nut', 'nut butter', 'oil', 'oilseed', 'other sauce', 'pasta', 'pasta sauce', 'pastry', 'pie', 'pizza', 'pizza sauce', 'plant-based meat', 'plant-based milk and cream', 'popcorn', 'pork', 'probiotic drink', 'protein drink', 'pudding', 'relish', 'rice', 'salad', 'salad dressing', 'salsa', 'salt and pepper', 'sandwich', 'sausage', 'shellfish', 'soft drink', 'soup', 'sparkling water', 'spice and herb', 'sport drink', 'spread', 'sugar', 'syrup', 'tea', 'turkey', 'vegetable', 'vinegar', 'water', 'white wine', 'wine and liquor', 'wrap', 'yogurt'

A.2 Food category classification evaluation

Table A.1: Food category classification evaluation for Word2Vec Google News model with 300 features. Shown are the training and validation accuracy of the 1 to 3 hidden layers Neural Networks for different learning rate.

Word2Vec

Learning Rate	0.01		0.003		0.001		0.0003		0.0001	
2-layers		Valid	Train	Valid	Train	Valid	Train	Valid	Train	Valid
Linear	0.9873	0.8107	0.9865	0.8247	0.9873	0.8270	0.9715	0.8247	0.8832	0.8099
Exp 0.98	0.9834	0.8287	0.9864	0.8378	0.9825	0.8285	0.8845	0.8116	0.7907	0.7592
Exp 0.96	0.9891	0.8343	0.9891	0.8371	0.9321	0.8260	0.8289	0.7869	0.7214	0.6930
3-layers										
Linear	0.9873	0.821	0.9896	0.8420	0.9880	0.8425	0.9627	0.8376	0.8339	0.7977
Exp 0.98	0.9880	0.8252	0.9886	0.8409	0.9748	0.8378	0.8288	0.7968	0.7268	0.7358
Exp 0.96	0.9848	0.8303	0.9840	0.8408	0.9094	0.8239	0.7586	0.7591	0.6339	0.6608
4-layers										
Linear	0.9795	0.7972	0.9884	0.8332	0.9889	0.8266	0.9832	0.8116	0.8284	0.7854
Exp 0.98	0.9858	0.8034	0.9878	0.8294	0.9844	0.8171	0.8479	0.7808	0.6430	0.6688
Exp 0.96	0.9831	0.8141	0.9853	0.8208	0.9469	0.8076	0.7161	0.7208	0.5486	0.6010

Table A.2: Food category classification evaluation for CLIP Vit-B/32 model with 512 features. Shown are the training and validation accuracy of the 1 to 3 hidden layers Neural Networks for different learning rate.

CLIP Vit-B-32

Learning Rate	0.01		0.003		0.001		0.0003		0.0001	
2-layers	Train	Valid	Train	Valid	Train	Valid	Train	Valid	Train	Valid
Linear	0.9912	0.8362	0.9907	0.8564	0.9897	0.8632	0.9906	0.8600	0.9891	0.8580
Exp 0.98	0.9906	0.8474	0.9897	0.8620	0.9900	0.8666	0.9897	0.8578	0.9356	0.8576
Exp 0.96	0.9918	0.8563	0.9918	0.8702	0.9918	0.8628	0.9618	0.8591	0.8907	0.8399
3-layers										
Linear	0.9870	0.8448	0.9917	0.8637	0.9920	0.8734	0.9916	0.8722	0.9618	0.8665
Exp 0.98	0.9912	0.8526	0.9918	0.8696	0.9918	0.8698	0.9700	0.8645	0.8735	0.8468
Exp 0.96	0.9905	0.8530	0.9915	0.8697	0.9884	0.8711	0.9074	0.8582	0.8226	0.8243
4-layers										
Linear	0.9753	0.8239	0.9913	0.8594	0.9915	0.8679	0.9911	0.8596	0.9759	0.8544
Exp 0.98	0.9868	0.8344	0.9925	0.8643	0.9915	0.8611	0.9800	0.8544	0.8408	0.8313
Exp 0.96	0.9892	0.8406	0.9907	0.8690	0.9891	0.8650	0.9057	0.8484	0.7724	0.7983

Table A.3: Food category classification evaluation for CLIP Vit-L/14 model with 768 features. Shown are the training and validation accuracy of the 1 to 3 hidden layers Neural Networks for different learning rate.

CLIP ViT-L/14

Learning Rate	0.01		0.003		0.001		0.0003		0.0001	
2-layers	Train	Valid	Train	Valid	Train	Valid	Train	Valid	Train	Valid
Linear	0.9911	0.8330	0.9914	0.8540	0.9902	0.8617	0.9904	0.8606	0.9912	0.8580
Exp 0.98	0.9912	0.8382	0.9902	0.8642	0.9882	0.8656	0.9908	0.8606	0.9785	0.8596
Exp 0.96	0.9918	0.8536	0.9918	0.8682	0.9918	0.8665	0.9897	0.8592	0.9399	0.8555
3-layers										
Linear	0.9779	0.8445	0.9910	0.8664	0.9913	0.8716	0.9918	0.8767	0.9836	0.8655
Exp 0.98	0.9906	0.8561	0.9921	0.8685	0.9910	0.8713	0.9868	0.8703	0.9227	0.8561
Exp 0.96	0.9893	0.8508	0.9916	0.8731	0.9906	0.8738	0.9559	0.8622	0.8712	0.8451
4-layers										
Linear	0.9582	0.8192	0.9906	0.8569	0.9914	0.8611	0.9916	0.8654	0.8703	0.8608
Exp 0.98	0.9857	0.8438	0.9919	0.8664	0.9916	0.8725	0.9894	0.8642	0.8977	0.8461
Exp 0.96	0.9855	0.8341	0.9914	0.8704	0.9909	0.8666	0.9535	0.8582	0.8215	0.8252

Table A.4: Food category classification evaluation for DINOv3 ViT-L/16 model with 2048 features model. Shown are the training and validation accuracy of the 1 to 3 hidden layers Neural Networks for different learning rate.

DINOv3 ViT-L/16

Learning Rate	0.01		0.003		0.001		0.0003		0.0001	
2-layers	Train	Valid	Train	Valid	Train	Valid	Train	Valid	Train	Valid
Linear	0.9580	0.7898	0.3991	0.8503	0.9909	0.8627	0.9900	0.8776	0.9903	0.8837
Exp 0.98	0.9794	0.8037	0.9907	0.8594	0.9903	0.8743	0.9884	0.8882	0.9908	0.8845
Exp 0.96	0.9860	0.8166	0.9918	0.8675	0.9918	0.8847	0.9918	0.8868	0.9918	0.8825
3-layers										
Linear	0.7194	0.7143	0.9889	0.8732	0.9922	0.8801	0.9922	0.8876	0.9918	0.8882
Exp 0.98	0.9038	0.7838	0.9907	0.8776	0.9927	0.8843	0.9913	0.8884	0.9891	0.8885
Exp 0.96	0.9079	0.7882	0.9918	0.8775	0.9915	0.8870	0.9895	0.8885	0.9730	0.8865
4-layers										
Linear	0.7130	0.6985	0.9890	0.8670	0.9918	0.8781	0.9915	0.8819	0.9911	0.8871
Exp 0.98	0.9388	0.7976	0.9910	0.8758	0.9916	0.8850	0.9916	0.8866	0.9847	0.8833
Exp 0.96	0.9247	0.8101	0.9909	0.8785	0.9923	0.8837	0.9891	0.8857	0.9453	0.8777

A.3 Food category classification evaluation with Data Augmentation

Table A.5: Food category classification evaluation with data augmentation for Word2Vec Google News model with 300 features. Shown are the training and validation accuracy of the 1 to 3 hidden layers Neural Networks for different learning rate.

Word2Vec

Learning Rate	0.01		0.003		0.001		0.0003		0.0001	
2-layers	Train	Valid	Train	Valid	Train	Valid	Train	Valid	Train	Valid
Linear	0.9857	0.8423	0.9789	0.8415	0.9478	0.8350	0.8534	0.8022	0.7364	0.7237
Exp 0.98	0.9803	0.8386	0.9488	0.8359	0.8653	0.8069	0.7396	0.7247	0.8666	0.5799
Exp 0.96	0.9668	0.8386	0.8984	0.8185	0.7953	0.7713	0.6490	0.6476	0.4023	0.4124
3-layers										
Linear	0.9686	0.8396	0.9717	0.8434	0.9553	0.8428	0.8205	0.7975	0.6753	0.6989
Exp 0.98	0.9603	0.8410	0.9491	0.8427	0.8285	0.8022	0.6582	0.6856	0.4845	0.5326
Exp 0.96	0.9428	0.8439	0.8832	0.8252	0.7276	0.7480	0.5539	0.5958	0.3027	0.3437
4-layers										
Linear	0.9572	0.8169	0.9787	0.8372	0.9671	0.8241	0.8847	0.7861	0.6461	0.6690
Exp 0.98	0.9447	0.8200	0.9612	0.8274	0.8835	0.7885	0.6456	0.6697	0.3613	0.4076
Exp 0.96	0.9194	0.8194	0.9292	0.8172	0.7827	0.7540	0.4547	0.4977	0.2410	0.2811

Table A.6: Food category classification evaluation with data augmentation for CLIP Vit-B/32 model with 512 features. Shown are the training and validation accuracy of the 1 to 3 hidden layers Neural Networks for different learning rate.

CLIP Vit-B-32

Learning Rate	0.01		0.003		0.001		0.0003		0.0001	
2-layers	Train	Valid	Train	Valid	Train	Valid	Train	Valid	Train	Valid
Linear	0.9878	0.8587	0.9901	0.8645	0.9907	0.8717	0.9827	0.8707	0.9499	0.8641
Exp 0.98	0.9895	0.8629	0.9909	0.8702	0.9871	0.8717	0.9540	0.8652	0.8977	0.8531
Exp 0.96	0.9890	0.8624	0.9885	0.8763	0.9767	0.8707	0.9186	0.8619	0.8620	0.8358
3-layers										
Linear	0.9529	0.8535	0.9875	0.8727	0.9862	0.8768	0.9796	0.8785	0.9236	0.8675
Exp 0.98	0.9722	0.8611	0.9858	0.8739	0.9821	0.8766	0.9299	0.8716	0.8426	0.8411
Exp 0.96	0.9692	0.8670	0.9819	0.8740	0.9676	0.8768	0.8729	0.8567	0.9009	0.8192
4-layers										
Linear	0.9377	0.8461	0.987	0.8632	0.9901	0.8675	0.9831	0.8706	0.9337	0.8608
Exp 0.98	0.9598	0.8540	0.9873	0.8702	0.9839	0.8704	0.9436	0.8701	0.8120	0.8299
Exp 0.96	0.9590	0.8562	0.9843	0.8683	0.9721	0.8732	0.8625	0.8457	0.7412	0.7875

Table A.7: Food category classification evaluation with data augmentation for CLIP Vit-L/14 model with 768 features. Shown are the training and validation accuracy of the 1 to 3 hidden layers Neural Networks for different learning rate.

CLIP ViT-L/14

Learning Rate	0.01		0.003		0.001		0.0003		0.0001	
2-layers	Train	Valid	Train	Valid	Train	Valid	Train	Valid	Train	Valid
Linear	0.9869	0.8563	0.9902	0.8689	0.9908	0.8748	0.9884	0.8720	0.9727	0.8690
Exp 0.98	0.9897	0.8619	0.9910	0.8727	0.9889	0.8775	0.9757	0.8706	0.9386	0.8647
Exp 0.96	0.9869	0.8634	0.9905	0.8757	0.9855	0.8776	0.9550	0.8660	0.9036	0.8564
3-layers										
Linear	0.9398	0.8546	0.9871	0.8729	0.9877	0.8769	0.9860	0.8771	0.9557	0.8787
Exp 0.98	0.9684	0.8632	0.9878	0.8745	0.9867	0.8782	0.9612	0.8752	0.8831	0.8549
Exp 0.96	0.9623	0.8646	0.9853	0.8763	0.9789	0.8796	0.9156	0.8690	0.8385	0.8420
4-layers										
Linear	0.9387	0.8461	0.9875	0.8694	0.9893	0.8736	0.9870	0.8725	0.9624	0.8740
Exp 0.98	0.9655	0.8618	0.9884	0.8715	0.9877	0.8748	0.9666	0.8706	0.8519	0.8473
Exp 0.96	0.9411	0.8522	0.9856	0.8706	0.9790	0.8736	0.9053	0.8611	0.7918	0.8205

Table A.8: Food category classification evaluation with data augmentation for DINOv3 ViT-L/16 model with 2048 features model. Shown are the training and validation accuracy of the 1 to 3 hidden layers Neural Networks for different learning rate.

DINOv3 ViT-L/16

Learning Rate	0.01		0.003		0.001		0.0003		0.0001	
2-layers	Train	Valid	Train	Valid	Train	Valid	Train	Valid	Train	Valid
Linear	0.9132	0.8321	0.9877	0.8680	0.9906	0.8743	0.9918	0.8814	0.9901	0.8854
Exp 0.98	0.9489	0.8457	0.9894	0.8690	0.9901	0.8806	0.9908	0.8852	0.9850	0.8862
Exp 0.96	0.9469	0.8434	0.9891	0.8739	0.9905	0.8834	0.9875	0.8885	0.9782	0.8841
3-layers										
Linear	0.6857	0.7231	0.9790	0.8771	0.9899	0.8801	0.9898	0.8856	0.9854	0.8894
Exp 0.98	0.8107	0.8004	0.9847	0.8809	0.9890	0.8866	0.9867	0.8912	0.9672	0.8927
Exp 0.96	0.8442	0.8206	0.9834	0.8839	0.9873	0.8887	0.9786	0.8907	0.9403	0.8873
4-layers										
Linear	0.7219	0.7297	0.9821	0.8763	0.9904	0.8782	0.9894	0.8820	0.9848	0.8912
Exp 0.98	0.8846	0.8247	0.9854	0.8763	0.9891	0.8839	0.9853	0.8918	0.9514	0.8874
Exp 0.96	0.8743	0.8254	0.9852	0.8810	0.9871	0.8856	0.9740	0.8910	0.9034	0.8794

A.4 Food category classification evaluation with Data Augmentation and Nutrition Information

Table A.9: Food category classification evaluation with data augmentation and nutrition information for Word2Vec Google News model with 300 features. Shown are the training and validation accuracy of the 1 to 3 hidden layers Neural Networks for different learning rate.

Word2Vec

Learning Rate	0.01		0.003		0.001		0.0003		0.0001	
2-layers	Train	Valid	Train	Valid	Train	Valid	Train	Valid	Train	Valid
Linear	0.9908	0.8516	0.9881	0.8536	0.9593	0.8460	0.8681	0.8183	0.7562	0.7516
Exp 0.98	0.9880	0.8527	0.9618	0.8493	0.8829	0.8243	0.7577	0.7516	0.5950	0.6190
Exp 0.96	0.9779	0.8559	0.9201	0.8414	0.8161	0.7920	0.6715	0.6831	0.4460	0.4696
3-layers										
Linear	0.9739	0.8527	0.9798	0.8568	0.9597	0.8536	0.8400	0.8190	0.7067	0.7352
Exp 0.98	0.9720	0.8531	0.9552	0.8554	0.8563	0.8254	0.7072	0.7349	0.5250	0.5822
Exp 0.96	0.9531	0.8513	0.9012	0.8399	0.7713	0.7851	0.6042	0.6600	0.3807	0.4441
4-layers										
Linear	0.9667	0.8369	0.9846	0.8502	0.9741	0.8432	0.8647	0.8152	0.6511	0.6995
Exp 0.98	0.9664	0.8389	0.9758	0.8469	0.8845	0.8220	0.6539	0.7045	0.4850	0.5503
Exp 0.96	0.9511	0.8375	0.9302	0.8399	0.7729	0.7704	0.5415	0.6066	0.3393	0.4079

Table A.10: Food category classification evaluation with data augmentation and nutrition information for CLIP Vit-B/32 model with 512 features. Shown are the training and validation accuracy of the 1 to 3 hidden layers Neural Networks for different learning rate.

CLIP Vit-B-32

Learning Rate	0.01		0.003		0.001		0.0003		0.0001	
2-layers	Train	Valid	Train	Valid	Train	Valid	Train	Valid	Train	Valid
Linear	0.9899	0.8595	0.9924	0.8684	0.9922	0.8771	0.9840	0.8731	0.9545	0.8661
Exp 0.98	0.9916	0.8623	0.9920	0.8776	0.9890	0.8745	0.9558	0.8676	0.9033	0.8575
Exp 0.96	0.9889	0.8645	0.9905	0.8776	0.9776	0.8730	0.9236	0.8655	0.8664	0.8406
3-layers										
Linear	0.9483	0.8555	0.9889	0.8725	0.9894	0.8795	0.9815	0.8831	0.9258	0.8726
Exp 0.98	0.9736	0.8650	0.9864	0.8754	0.9843	0.8804	0.9295	0.8701	0.8461	0.8476
Exp 0.96	0.9698	0.8639	0.9837	0.8780	0.9663	0.8794	0.8771	0.8603	0.7992	0.8204
4-layers										
Linear	0.9379	0.8506	0.9885	0.8665	0.9898	0.8748	0.9844	0.8768	0.9308	0.8693
Exp 0.98	0.9586	0.8576	0.9879	0.8724	0.9852	0.8712	0.9401	0.8712	0.8118	0.8315
Exp 0.96	0.9582	0.8586	0.9852	0.8708	0.9729	0.8734	0.8579	0.8490	0.7472	0.8006

Table A.11: Food category classification evaluation with data augmentation and nutrition information for CLIP Vit-L/14 model with 768 features. Shown are the training and validation accuracy of the 1 to 3 hidden layers Neural Networks for different learning rate.

CLIP ViT-L/14

Learning Rate	0.01		0.003		0.001		0.0003		0.0001	
2-layers	Train	Valid	Train	Valid	Train	Valid	Train	Valid	Train	Valid
Linear	0.9875	0.8585	0.9919	0.8647	0.9918	0.8781	0.9887	0.8766	0.9739	0.8701
Exp 0.98	0.9893	0.8664	0.9920	0.8778	0.9904	0.8780	0.9761	0.8702	0.9407	0.8637
Exp 0.96	0.9882	0.8623	0.9906	0.8797	0.9867	0.8758	0.8568	0.8690	0.9083	0.8587
3-layers										
Linear	0.9428	0.8549	0.9877	0.8740	0.9886	0.8831	0.9870	0.8805	0.9556	0.8771
Exp 0.98	0.9683	0.8638	0.9885	0.8752	0.9875	0.8827	0.9600	0.8786	0.8858	0.8594
Exp 0.96	0.9690	0.8670	0.9865	0.8758	0.9792	0.8827	0.9168	0.8701	0.8430	0.8448
4-layers										
Linear	0.9323	0.8462	0.9884	0.8716	0.9905	0.8775	0.9881	0.8776	0.9644	0.8773
Exp 0.98	0.9683	0.8580	0.9884	0.8720	0.9877	0.8786	0.9671	0.8783	0.8541	0.8494
Exp 0.96	0.9592	0.8594	0.9862	0.8759	0.9818	0.8762	0.9090	0.8638	0.7955	0.8215

Table A.12: Food category classification evaluation with data augmentation and nutrition information for DINOv3 ViT-L/16 model with 2048 features model. Shown are the training and validation accuracy of the 1 to 3 hidden layers Neural Networks for different learning rate.

DINOv3 ViT-L/16

Learning Rate	0.01		0.003		0.001		0.0003		0.0001	
2-layers	Train	Valid	Train	Valid	Train	Valid	Train	Valid	Train	Valid
Linear	0.9151	0.8357	0.9880	0.8698	0.9904	0.8757	0.9915	0.8825	0.9902	0.8868
Exp 0.98	0.9492	0.8439	0.9893	0.8740	0.9911	0.8848	0.9902	0.8833	0.9855	0.8876
Exp 0.96	0.9511	0.8546	0.9886	0.8808	0.9898	0.8842	0.9877	0.8871	0.9786	0.8856
3-layers										
Linear	0.6663	0.7080	0.9795	0.8772	0.9893	0.8817	0.9883	0.8875	0.9852	0.8931
Exp 0.98	0.8159	0.8025	0.9852	0.8796	0.9895	0.8859	0.9864	0.8910	0.9671	0.8917
Exp 0.96	0.8349	0.8090	0.9849	0.8827	0.9866	0.8887	0.9784	0.8924	0.9357	0.8878
4-layers										
Linear	0.7222	0.7322	0.9829	0.8778	0.9900	0.8800	0.9901	0.8829	0.9859	0.8866
Exp 0.98	0.8765	0.8236	0.9861	0.8786	0.9902	0.8818	0.9856	0.8898	0.9540	0.8878
Exp 0.96	0.8704	0.8210	0.9835	0.8824	0.9861	0.8884	0.9754	0.8893	0.9050	0.8834

REFERENCES

[1] Amazon. *Welcome to AWS Documentation*. https://docs.aws.amazon.com/. Online; accessed 30-April-2026. 2026.

[2] Emanuele Bastianelli et al. "SLURP: A Spoken Language Understanding Resource Package." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber et al. Online: Association for Computational Linguistics, Nov. 2020, pp. 7252–7262. DOI: 10.18653/v1/2020.emnlp-main.588. URL: https://aclanthology.org/2020.emnlp-main.588.

[3] Qian Chen, Zhu Zhuo, and Wen Wang. "BERT for Joint Intent Classification and Slot Filling." In: *CoRR* abs/1902.10909 (2019). arXiv: 1902.10909. URL: http://arxiv.org/abs/1902.10909.

[4] Alice Coucke et al. *Snips Voice Platform: an embedded Spoken Language Understanding system for private-by-design voice interfaces*. 2018. arXiv: 1805.10190 [cs.CL]. URL: https://arxiv.org/abs/1805.10190.

[5] Jacob Devlin, Ming-Wei Chang, et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[6] Jacob Devlin, Ming-Wei Chang, et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

[7] OpenJS Foundation. *Express, Fast, unopinionated, minimalist web framework for Node.js*. https://expressjs.com/. Online; accessed 30-April-2026. 2026.

[8] OpenJS Foundation. *Node.js v25.9.0 documentation*. https://nodejs.org/docs/latest/api/. Online; accessed 30-April-2026. 2026.

[9] Python Software Foundation. *Requests: HTTP for Humans*. https://pypi.org/project/requests/. [Online; accessed 06-March-2025]. 2024.

[10] Chih-Wen Goo et al. "Slot-Gated Modeling for Joint Slot Filling and Intent Prediction." In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Ed. by Marilyn Walker, Heng Ji, and Amanda Stent. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 753–757. DOI: 10.18653/v1/N18-2118. URL: https://aclanthology.org/N18-2118.

[11] Google. *Databases, Google Cloud Documentation*. https://docs.cloud.google.com/docs/databases. Online; accessed 30-April-2026. 2026.

[12] The PostgreSQL Global Development Group. *PostgreSQL: The World's Most Advanced Open Source Relational Database*. https://www.postgresql.org/. Online; accessed 30-April-2026. 2026.

[13] Richard Andrew Harrington et al. "Nutrient composition databases in the age of big data: foodDB, a comprehensive, real-time database infrastructure." In: *BMJ Open* 9.6 (2019). ISSN: 2044-6055. DOI: 10.1136/bmjopen-2018-026652. eprint: https://bmjopen.bmj.com/content/9/6/e026652.full.pdf. URL: https://bmjopen.bmj.com/content/9/6/e026652.

[14] Rohana N. Haththotuwa, Chandrika N. Wijeyaratne, and Upul Senarath. "Chapter 1 - Worldwide epidemic of obesity." In: *Obesity and Obstetrics (Second Edition)*. Ed. by Tahir A. Mahmood, Sabaratnam Arulkumaran, and Frank A. Chervenak. Second Edition. Elsevier, 2020, pp. 3–8. ISBN: 978-0-12-817921-5. DOI: https://doi.org/10.1016/B978-0-12-817921-5.00001-1. URL: https://www.sciencedirect.com/science/article/pii/B9780128179215000011.

[15] Charles T. Hemphill, John J. Godfrey, and George R. Doddington. "The ATIS Spoken Language Systems Pilot Corpus." In: *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*. 1990. URL: https://aclanthology.org/H90-1021.

[16] Instacart. *Instacart retailer website*. https://www.instacart.com/. [Online; accessed 06-March-2025]. 2025.

[17] Cijo Jose et al. *DINOv2 Meets Text: A Unified Framework for Image- and Pixel-Level Vision-Language Alignment*. 2024. arXiv: 2412.16334 [cs.CV]. URL: https://arxiv.org/abs/2412.16334.

[18] Armand Joulin et al. *FastText.zip: Compressing text classification models*. 2016. arXiv: 1612.03651 [cs.CL]. URL: https://arxiv.org/abs/1612.03651.

[19] Bing Liu and Ian R. Lane. "Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling." In: *CoRR* abs/1609.01454 (2016). arXiv: 1609.01454. URL: http://arxiv.org/abs/1609.01454.

[20] Vukosi Marivate and Tshephisho Sefara. "Improving short text classification through global augmentation methods." In: *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*. Springer. 2020, pp. 385–399.

[21] Grégoire Mesnil et al. "Using recurrent neural networks for slot filling in spoken language understanding." In: *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* 23.3 (Mar. 2015), pp. 530–539. ISSN: 2329-9290.

[22] Microsoft. *Azure documentation, Microsoft Learn*. https://learn.microsoft.com/en-us/azure/?product=popular. Online; accessed 30-April-2026. 2026.

[23] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL]. URL: https://arxiv.org/abs/1301.3781.

[24] Anthony Moi and Nicolas Patry. *HuggingFace's Tokenizers*. Version 0.13.4. Apr. 2023. URL: https://github.com/huggingface/tokenizers.

[25] Henry B. Moss et al. "BOSS: Bayesian Optimization over String Spaces." In: *CoRR* abs/2010.00979 (2020). arXiv: 2010.00979. URL: https://arxiv.org/abs/2010.00979.

[26] Baiju Muthukadan. *Selenium with Python*. https://selenium-python.readthedocs.io/. [Online; accessed 06-March-2025]. 2024.

[27] OpenAI. *Models of GPT-4*. https://platform.openai.com/docs/models/gpt-4. [Online; accessed 06-March-2025]. 2024.

[28] OpenAI. *Models of GPT-4 omni*. https://platform.openai.com/docs/models/gpt-4o. [Online; accessed 06-March-2025]. 2024.

[29] Maxime Oquab et al. *DINOv2: Learning Robust Visual Features without Supervision*. 2024. arXiv: 2304.07193 [cs.CV]. URL: https://arxiv.org/abs/2304.07193.

[30] Oracle. *MySQL*. https://www.mysql.com/. Online; accessed 30-April-2026. 2026.

[31] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162/.

[32] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV]. URL: https://arxiv.org/abs/2103.00020.

[33] Mehrdad Rafiepour and Javad Salimi Sartakhti. "CTRAN: CNN-Transformer-based network for natural language understanding." In: *Engineering Applications of Artificial Intelligence* 126 (Nov. 2023), p. 107013. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2023.107013. URL: http://dx.doi.org/10.1016/j.engappai.2023.107013.

[34] Radim Řehůřek and Petr Sojka. "Software Framework for Topic Modelling with Large Corpora." English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50.

[35] Render. *Render, The cloud for builders*. https://render.com/. Online; accessed 30-April-2026. 2026.

[36] Oriane Siméoni et al. *DINOv3*. 2025. arXiv: 2508.10104 [cs.CV]. URL: https://arxiv.org/abs/2508.10104.

[37] TraderJoes. *Trader Joe's retailer website*. https://www.traderjoes.com/. [Online; accessed 06-March-2025]. 2025.

[38] USDA. *U.S. Obesity Rate Changes Differ for Rural and Urban Areas, as Well as Across Regions*. https://www.ers.usda.gov/amber-waves/2025/august/us-obesity-rate-changes-differ-for-rural-and-urban-areas-as-well-as-across-regions. Online; accessed 05-November-2025. 2025.

[39] Ngoc Thang Vu. "Sequential Convolutional Neural Networks for Slot Filling in Spoken Language Understanding." In: *Interspeech 2016*. 2016, pp. 3250–3254. DOI: 10.21437/Interspeech.2016-395.

[40] Yu Wang, Yilin Shen, and Hongxia Jin. *A Bi-model based RNN Semantic Frame Parsing Model for Intent Detection and Slot Filling*. 2018. arXiv: 1812.10235 [cs.CL]. URL: https://arxiv.org/abs/1812.10235.

[41] Jason Wei and Kai Zou. *EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks*. 2019. arXiv: 1901.11196 [cs.CL]. URL: https://arxiv.org/abs/1901.11196.

[42] Puyang Xu and Ruhi Sarikaya. "Convolutional neural network based triangular CRF for joint intent detection and slot filling." In: *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. 2013, pp. 78–83. DOI: 10.1109/ASRU.2013.6707709.

BIOGRAPHICAL SKETCH

The author graduated with a Bachelor of Business Administration in Statistics from National Cheng Kung University (NCKU) in Tainan City, Taiwan, in June 2018. Following graduation, the author completed a research internship at the NCKU Institute of Molecular Medicine, where the author focused on improving the testing power of Normalized Chromosome Values to detect genomic variations. From March 2019 to April 2020, the author served as a Research Assistant for a Ministry of Science and Technology Program at the NCKU Institute of Molecular Medicine. During this tenure, the author contributed to Noninvasive Prenatal Testing (NIPT) projects by estimating fetal cell-free DNA fractions and detecting risks of copy number variations. The research interests in this stage bridge statistical methodology and bioinformatics, with a particular focus on high-throughput data analysis, multiomics, and computational data science. The author optimized operational taxonomic unit analysis workflows using R and Linux, and presented their work on single-nucleotide polymorphism-based fetal fraction estimation at the Multiomics and Precision Medicine Joint Conference. Additionally, the author served as a Teaching Assistant for specialized workshops on Exome-Seq and Next Generation Sequencing data analysis across Taiwan, instructing participants in Linux environments and advanced bioinformatics tools such as Samtools, GATK, CNVkit, and Cn.mops. The author is currently pursuing a Doctor of Philosophy in Statistics at Florida State University in Tallahassee, Florida.