applied sciences

MDPI

Article

Enhancing Food Ingredient Named-Entity Recognition with Recurrent Network-Based Ensemble (RNE) Model

Kokoy Siti Komariah [1] and Bong-Kee Sin [1,2,*]

[1] Department of AI Convergence, Pukyong National University, Busan 48513, Korea
[2] Division of Computer Engineering, Pukyong National University, Busan 48513, Korea
* Correspondence: bkshin@pknu.ac.kr

Abstract: Food recipe sharing sites are becoming increasingly popular among people who want to learn how to cook or plan their menu. Through online food recipes, individuals can select ingredients that suit their lifestyle and health condition. Information from online food recipes is useful in developing food-related systems such as recommendations and health care systems. However, the information from online recipes is often unstructured. One way of extracting such information into a well-structured format is the technique called named-entity recognition (NER), which is the process of identifying keywords and phrases in the text and classifying them into a set of predetermined categories, such as location, persons, time, and others. We present a food ingredient named-entity recognition model called RNE (recurrent network-based ensemble methods) to extract the entities from the online recipe. RNE is an ensemble-learning framework using recurrent network models such as RNN, GRU, and LSTM. These models are trained independently on the same dataset and combined to produce better predictions in extracting food entities such as ingredient names, products, units, quantities, and states for each ingredient in a recipe. The experimental findings demonstrate that the proposed model achieves predictions with an F1 score of 96.09% and outperforms all individual models by 0.2% to 0.5% in percentage points. This result indicates that RNE can extract information from food recipes better than a single model. In addition, this information extracted by RNE can be used to support various information systems related to food.

Keywords: deep learning; ensemble method; food information extraction; named-entity recognition; recurrent models

Citation: Komariah, K.S.; Sin, B.-K. Enhancing Food Ingredient Named-Entity Recognition with Recurrent Network-Based Ensemble (RNE) Model. *Appl. Sci.* **2022**, *12*, 10310. https://doi.org/10.3390/app122010310

Academic Editor: Giancarlo Mauri

Received: 24 August 2022
Accepted: 10 October 2022
Published: 13 October 2022

1. Introduction

In the not-so-distant past, personal collections of recipes were kept in binders, notes, and magazine clippings. Nowadays, the internet and digitization are changing these practices. People are starting to post their favorite recipes on recipe sharing sites. These recipe sharing sites introduce people worldwide to new dishes suitable for their lifestyle and well-being. Thousands of recipes are available online for individuals who enjoy cooking at home or planning their menus. In addition, this food-related information is useful for developing health and health care-related systems, such as personalized recommendation systems for individual diets [1,2], calculating nutrient levels in foods to prevent malnutrition [3], and identifying alternative ingredients for individuals with food allergies [4,5].

However, the information from online recipes is often given in the format of unstructured text. One of the most basic tools to extract such information into a well-structured format is the technique called named-entity recognition (NER). NER enables us to recognize and extract essential pieces of information from text, such as names, locations, organizations, and so on [5,6]. NER can also help various enterprises and organizations discover insights from massive unstructured datasets. Furthermore, it can be useful for analyzing relations that exist between these entities. In the case of food and meal processing terms, NER can help find some hidden relations between food entities and disease entities for food allergies studies.

Numerous studies have been conducted on NER applications in various fields [7,8]. Most of this research employs single classifiers such as CRF, HMM, LSTM, Bi-LSTM, BERT, and their variations, and tried to combine them in an end-to-end manner like Bi-LSTM-CRF. In doing so, they failed to benefit from the diversity of existing NER algorithms. On the other hand, it is a well-known fact that algorithms with different strengths and weaknesses can be coupled in various ways to build a model that surpasses the best individual algorithm. This type of learning is referred to as the ensemble method. As shown in Table 1, the majority of ensemble methods for NER employ voting techniques, whereas only a few employ stacking and concatenation techniques. However, none of these studies has yet incorporated it as a solution to the issue of extracting food-related entities.

Table 1. Comparative analysis of previous studies on NER using ensemble method.

Study	Ensemble Scheme	Classifiers	Dataset	Performance Metrics
[9]	Voting	MLP, ABM1, J48	Reuters corpus	Recall, Precision, Error rate, MCC, F1 score
[10]	Weighted Voting	ME, CRF, SVM	Bengali News corpus, NERSSEAL, CoNLL-2003	F1 score
[11]	Concatenation	Neural Networks	OKE2016, AIDA/CoNLL, NexGenTV corpus	Recall, Precision, F1 score
[12]	Arbitration Rules, Stacked Generalization, Cascade Generalization	Generalized Winnow, ME, SVM, CRF	GENIA, JNLPBA	Recall, Precision, F1 score
[13]	Voting	5 NER systems: Stanford NER, NER-Tagger, Edinburgh Geoparser, spaCy NER, Polyglot	The Marry Hamilton Papers, The Samuel Hartlib Papers	Recall, Precision, F1 score
[14]	Majority Voting	BERT, CNN, CamemBERT, CamemBERT-bio, XL-Net, RoBERTa, BioBERT, Bio + ClinicalBERT, PubMedBERT, BioMed RoBERTa	ChEMU 2020, DEFT 2020, WNUT 2020	F1 score
[15]	Majority Voting, Stacking	SVM, CRF, ME	i2b2 2010 corpus	Recall, Precision, F1 score
[16]	Majority Voting	BERT-base-cased, BERT-based-uncased, CNN	ChEMU NER Task	Recall, Precision, F1 score
[17]	Plurality Voting, Weighted Voting	HMM, CRF, MEM, BiLSTM	Private (Authors private data)	F1 score

Therefore, we propose the recurrent network-based ensemble (RNE) method, an ensemble approach involving deep recurrent neural networks (RNNs) to extract food entities from unstructured textual data. The concept behind utilizing RNNs for NER is to make use of sequential information. NER is a classification problem at the token level. Thus, the ability to understand sentence context from beginning to end is needed for more accurate predictions. Recurrent neural networks (RNNs), along with their variants such as gated recurrent unit (GRU) and long-short-term memory (LSTM), have shown remarkable results in sequential data modelling [8,18]. For example, NER architectures that integrated and employed RNNs as a feature extractor for word context or word characters have shown to be powerful NER systems that require little domain-specific knowledge or resources [18–20].

Furthermore, a bidirectional RNN has been proven to efficiently utilize future information (via the backward state) as well as past information (via the forward state) for a certain period of time [21]. On the other hand, ensemble learning is a method to leverage consensus in prediction by combining relevant aspects of different models. Since the ensemble

methods help reduce the variance of prediction errors, the final framework is expected to be more robust than the individual models that comprise the ensemble [22]. Therefore, we expect that an RNE that leverages the advantages of deep RNNs and ensemble learning can provide a useful framework for NER in identifying food ingredient entities. We trained three independent deep learning models (RNN, GRU, and LSTM) on the same dataset and fused their prediction with the ensemble concatenation [11] approach for a final prediction.

Our main contributions in this paper are summarized as follows:

- Proposing a novel approach to extract food-related entities using an ensemble method with recurrent networks models such as RNN, GRU, and LSTM. To the best of the authors' knowledge, this is the first work to explore a model that benefits from a deep learning-based ensemble method applied to a food-related NER task.
- Preparing a carefully annotated dataset for food ingredient entities, namely the FINER dataset.
- Achieving better results on the proposed NER task approach compared to single models for food-related NER.

In this paper, we present a robust strategy for extracting food ingredient information from recipe text. The rest of the chapter is organized as follows: Section 2 presents an overview of the related works; Section 3 defines the proposed method by constructing the food ingredient dataset, deep learning classifiers, and finally our proposed method RNE; Section 4 analyzes and discusses the results of the proposed method; and Section 5 concludes the study.

2. Related Works

This section will briefly review the literature on developing NER for food information extraction, including several studies using the recurrent networks model and ensemble methods in tackling NER problems.

2.1. Food-Related NER

NER tools and frameworks employ a wide range of approaches that can be divided into three categories: dictionary-based, rule-based, and machine learning-based [7]. There are a few studies on the NER problem in the food domain. In the ruled-based approach, there are FoodIE [23] and drNER [24]. In addition, several databases for food-related NER have also been developed, such as RecipeDB [25] and FoodBase [26]. FoodBase raw data were obtained from the Allrecipes website, while RecipeDB was from Allrecipes and Food.com websites. Following the creation of the FoodBase corpus, the latest study in [5] utilized this dataset for evaluating four different NER methods, namely FoodIE, NCBO (SNOMED CT), NCBO (OntoFood), and NCBO (FoodON). Furthermore, to improve the FoodBase corpus performance, BuTTER [27] and FoodNER [28] were proposed. BuTTER employs bidirectional long short-term memory (BiLSTM) and conditional random fields (CRFs), while the latter utilizes a bidirectional encoder representation from the transformers (BERTs) model to extract food entities.

2.2. Recurrent Networks Model for NER

The emergence of deep learning for NLP has brought a new research paradigm. Several studies on deep learning-related models and approaches have been used for various NLP tasks [29]. One of them is recurrent neural networks. RNNs and their variants, such as GRU and LSTM, have exhibited impressive results in modeling sequential data [8,18]. Particularly, bidirectional RNNs make efficient use of past information (through forward states) as well as future information (via backward states) for a given period [30]. The first study employing RNNs for NER tasks was conducted by [30] utilizing the LSTM–CRF architecture. Following this, several additional studies also investigated RNNs for sequence-labeling problems. The authors of [20,31–33] used LSTM, whereas the authors of [34–37] used GRU in completing their NER tasks. In addition, the RNN-based model, the authors of [38] employ RNNs for nested NER problems and the authors of [39] use RNNs

for NER in Chinese electronic medical records. In the case of food information extraction, MenuNER [1] and BuTTER [27] were developed using Bi-LSTM-CRF.

2.3. Ensemble Methods for NER

The objective of an ensemble learning algorithm is to build a classifier with a high predictive performance by combining the predictions of multiple base classifiers. Different approaches have been developed to efficiently combine base classifiers [40,41]. The most common approaches include voting, bagging, boosting, blending, and stacking [22,42,43]. In the NER task, several research studies have been carried out using the ensemble method. Table 1 provides a summary of these studies. Among the ensemble strategy, voting has been a method of choice for most of the NER studies [9,10,13–17]. Voting is an ensemble method that incorporates the performance of various classifiers, in which the class with the most votes is selected from the base classifiers [41]. Some studies in Table 1 implement different strategies such as stacking [15], concatenation [11], and other approaches [12]. However, none of these studies have yet been utilized for extracting food-related entities. Thus, in this study, we proposed a recurrent networks-based ensemble method for extracting food-related entities, namely RNE. We will explain RNE in detail in the next section of the proposed methods.

3. Proposed Methods

This section explains the steps and materials required to develop the proposed RNE method as follows: (1) The construction of the dataset will be shown first to demonstrate how we generate and produce our dataset using textual data collected from the Allrecipes website. In the following step, we show how the data is preprocessed to be suitable input for our model. (2) As a building block of our ensemble model, we introduce and explain three variants of recurrent networks: RNN, GRU, and LSTM. (3) The recur-rent network-based ensemble (RNE) method is presented to explain the use of ensemble learning methods using multiple recurrent-based networks in identifying ingredient entities in a recipe text.

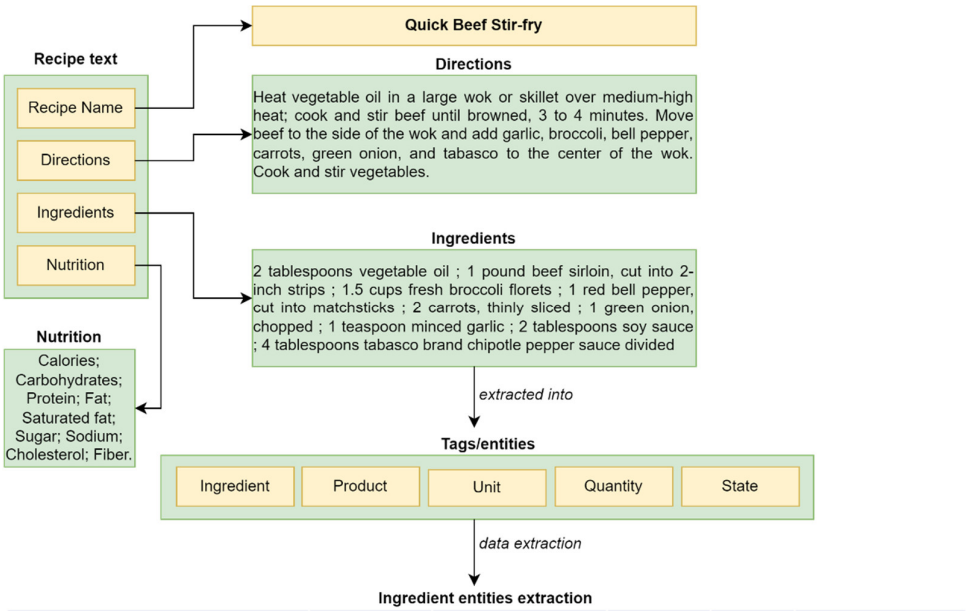
3.1. Dataset Construction

The dataset used in this study was obtained from the Allrecipes website [44]. It contains 64,782 recipe data elements from all food categories. As shown in Figure 1, the data consist of recipe names, ingredients, directions, and nutritional information. This study uses the ingredient section information and extracts ingredient entities and their attributes, such as ingredient name, product, unit, quantity, and state. In creating the dataset, several NLP libraries, including SpaCy [45], NLTK [46], and Doccano annotation tools [47], were used. The detailed data construction procedure shown in Figure 2 consists of four steps, as follows:

(a) *Data preparation.* We start with raw texts obtained by scraping the Allrecipes website and arranging it so that the format and structure of the data match the desired input. Several pre-processing steps are taken, such as: tokenizing the text, converting it into lowercase, removing all special characters, white spaces, stop words, punctuation, and lemmatization. We also define additional rules as follows:

- Since our dataset is a list of ingredients, stop words and punctuations are not always meaningless to the intention of the text but can potentially help interpret entities. Thus, we have developed custom stop words and punctuation lists. For example, the sentence "*1 (2 ounces) package butter*" means 1 package of butter equals 2 ounces. Although the word "*2 ounces*" is in brackets, we keep the parentheses because they carry clues for converting the amount of ingredient into standard units.
- We standardized the measurements for units and quantities. For example, in units, we convert all abbreviations into their original form, such as "*tbsp*" becomes "*tablespoon*"; and in quantities, we convert all numbers from fractions into a decimal, such as "1/2" becomes "*0.5*".

- In addition, to simplify the extraction process, we segment each phrase in the ingredients section into individual phrases. Figure 1 illustrates the process of extracting the ingredient entity from the text recipe. It can be observed that the recipe text consists of multiple sections, such as recipe name, directions, ingredients, and nutrition. In our experiment, only the ingredients section was utilized. This section contains a list of ingredients required for a particular recipe. Each list is then divided into ingredient phrases, as depicted in Figure 3. Finally, the resulting dataset consists of 181,970 phrases in total after the pre-processing step.



Ingredient phrase	Ingredient	Product	Unit	Quantity	State
2 tablespoons vegetable oil	vegetable oil		tablespoons	2	
1 pound beef sirloin, cut into 2-inch strips	beef sirloin		pound	1	cut into 2-inch strips
1.5 cups fresh broccoli florets	broccoli florets		cups	1.5	fresh
1 red bell pepper, cut into matchsticks	red bell pepper			1	cut into matchsticks
2 carrots, thinly sliced	carrots			2	thinly sliced
1 green onion, chopped	green onion			1	chopped
1 teaspoon minced garlic	garlic		tablespoons	1	minced
2 tablespoons soy sauce	soy sauce		tablespoons	2	
4 tablespoons tabasco brand chipotle pepper sauce divided		tabasco brand chipotle pepper sauce	tablespoons	4	divided

Figure 1. Ingredient data extraction workflow from Allrecipes data.

(b) *Manual data annotation.* We start with a small set of 2000 instances that have been manually labeled and divide it in half. One is for initial training, and the other for evaluation. The initial training set is used to develop a baseline NER annotator and the evaluation set is kept for the final evaluation of the complete dataset.

(c) *Model training and automatic labeling.* In this step, we create baseline models using the initial training set from the previous stage. We develop a semi-supervised method called SMPT (semi-supervised multi-model prediction technique) to annotate and create our dataset. SMPT adopts the self-training concept that builds on pre-trained models in the iterative data labeling process [48]. After a number of repetitions, the resulting dataset is re-appended to the training set, which grows rapidly to the desired size. Figure 4 depicts the architecture of the SMPT method. Initially, we create a small dataset of manually labeled instances and then train a set of base classifiers built on pretrained language models from spaCy, BERT, and DistilBERT. Through the SMPT, we extract all entities in the ingredient section of the text and then annotate them into five entities: ingredient, product, quantity, unit, and state.

These are applied to unlabeled sets to obtain additional named entities using a voting scheme for improved prediction. The resulting models are then employed to predict the remaining unlabeled instances. Once we have generated new labeled data, some of it is promoted into the existing labeled set. This procedure is repeated until we run out of unlabeled data.

(d) *Dataset evaluation.* After several repetitions, we end up with the Food Ingredient NER (FINER) dataset. We indirectly evaluate the dataset's quality using a set of classifiers such as CRF, Bi-LSTM, and BERT. We evaluate their performance on the evaluation set that has been reserved.

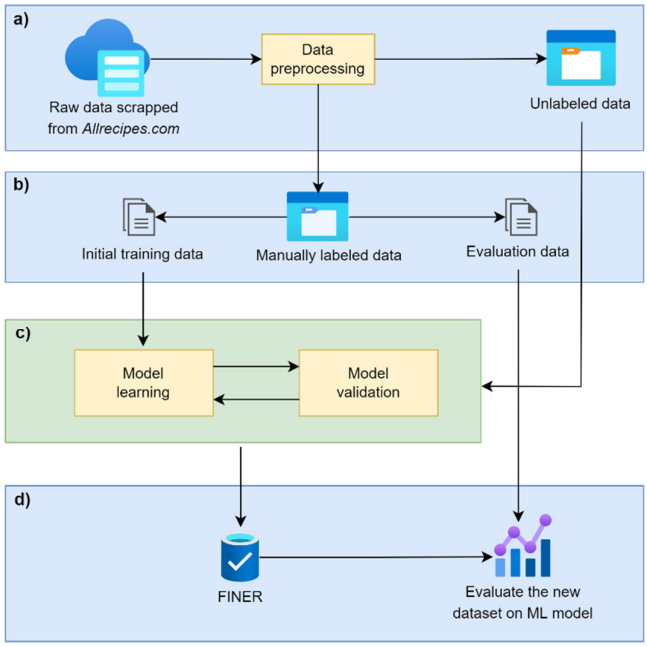


Figure 2. Overall flowchart for constructing the FINER dataset. The construction process including (**a**) data preparation; (**b**) manual data annotation; (**c**) model training and automatic labeling; and, (**d**) dataset evaluation.

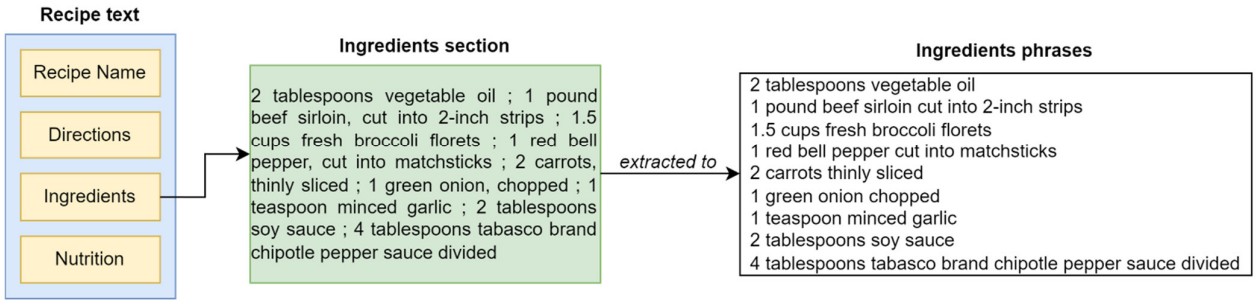


Figure 3. Segmentation of ingredient lists into individual phrases.

We call the resulting dataset from the above procedure Food Ingredient NER, or FINER for short. This dataset consists of 1,397,960 words in total with 181,970 sentences; the dataset's detailed specification is shown in Table 2. Five different entity classes are defined, each corresponding to an entity tag shown in Table 3. To chunk the entity word, we employed the IOB2 format [6,49]. The IOB2 format is similar to the IOB format, except for the addition of the B-tag at the beginning of each chunk (i.e., all chunks start with a B-tag). Table 4 depicts a detailed explanation of the IOB format. In this format, a tag is prefixed by either B, I, or O, indicating the position within the entity. [*B-tag*] indicates that the tag is the beginning of a chunk, [*I-tag*] indicates that the tag is inside a chunk, and [*O*] indicates that a token is not a chunk. The "tag" shall be replaced with a named entity label

such as [*ING*] for ingredients in our data. The complete distribution of the dataset over tags is further presented in Table 5.

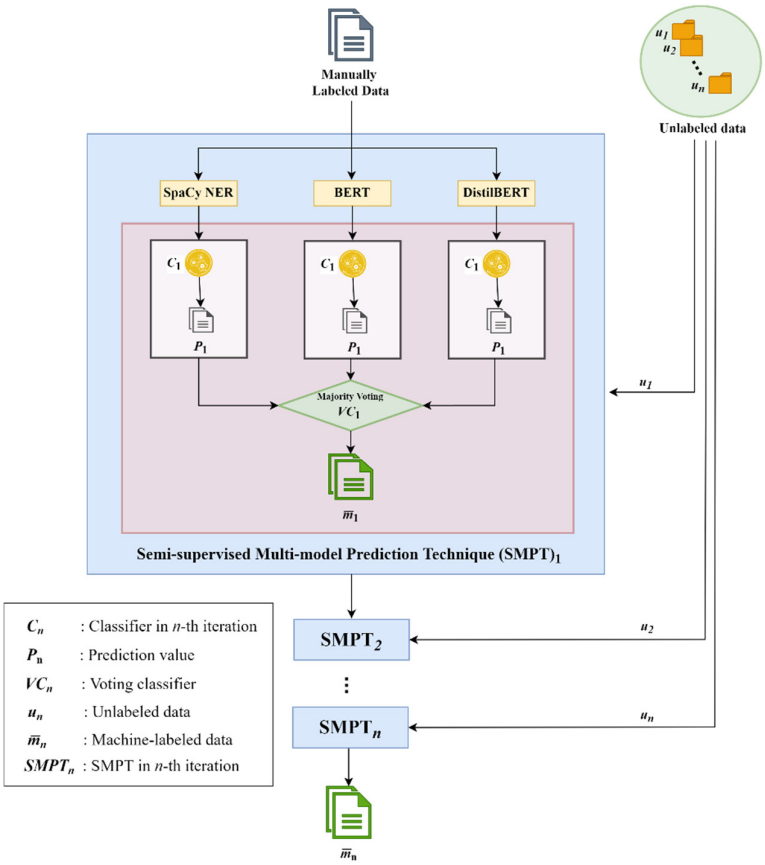


Figure 4. SMPT method for ingredient-named entity data annotation.

Table 2. The dataset specification details.

Total number of words	1,397,960
Total number of sentences	181,970
Total number of entities (without O tags)	1,177,660
Total number of tags (without O tags)	10

Table 3. Entity classes with corresponding acronyms and examples.

Class	Description	Example
INGREDIENT	Name of the food or ingredient.	Carrots, garlic, vegetable oil, etc.
PRODUCT	Food or ingredient from specific brand mention.	Tabasco-brand chipotle pepper sauce, Archer farms dark chocolate hot cocoa mix, etc.
QUANTITY	Measurement unit.	Gram, pound, tablespoon, etc.
STATE	Processing state of the food or ingredient.	Minced, chopped, cut into 2-inch strips, etc.
UNIT	Measurements of the food or ingredient associated with the unit.	1 1/2, 25, 0.5, etc.

Table 4. IOB tagging scheme.

Tag	Definition
B (Begin)	Prefix before a tag indicates that the tag is the beginning of a chunk.
I (Inside)	Prefix before a tag indicates that the tag is inside a chunk.
O (Outside)	Tag indicates that a token belongs to a non-chunk (outside).

Table 5. Distribution of the dataset for named entities.

Named-Entity Type	Count	Ratio (%)
B-INGREDIENT	210,082	15.03
B-PRODUCT	17,325	1.24
B-QUANTITY	209,867	15.01
B-STATE	135,315	9.68
B-UNIT	174,993	12.52
I-INGREDIENT	240,436	17.20
I-PRODUCT	55,212	3.95
I-QUANTITY	1919	0.14
I-STATE	130,158	9.31
I-UNIT	2353	0.17
O (outside or non-entity chunk)	220,300	15.76
Total	1,397,960	100

3.2. Recurrent Networks Classifiers

This study employs three recurrent deep learning models: RNN, LSTM, and GRU. We present an ensemble learning model to solve a food ingredient NER problem by combining all three models into an ensemble model to obtain an improved performance. Following is a detailed explanation of each model.

3.2.1. Recurrent Neural Networks

A recurrent neural network (RNN) [50] is a type of artificial neural network that can store information about preceding inputs in a sequence in its memory. Figure 5 depicts the structure of basic RNNs. In RNNs, each word of the input sequence $x_1, x_2, \ldots, x_n$ turns into vector form $\hat{y}_t$ by using the following equations:

$$h_t = H(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (1)$$

$$\hat{y}_t = W_{hy}h_t + b_y \quad (2)$$

where W_{xh}, W_{hh}, and W_{hy} represent the weight matrices, h_t is the vector of hidden states that capture the information at the time slice $t (t = 1, 2, \ldots, T)$, b represent biases, and H is the collection of activation functions for the hidden layer. The basic RNN with unidirectional flow passes the data from left to right. Shared parameters are used for each time step. In Figure 4, W_{xh}, W_{hh}, and W_{hy} are the same for each time step. When generating the prediction at time t, it uses not only the current input x_t at time t but also information from prior input at time $t-1$ through the activation parameter H and weights W, which passes from the previously hidden layer to the current hidden layer. One problem with this RNN is that it can only make predictions based on the information that comes before it. In that case, RNN uses information from earlier in the sequence to make a prediction at a certain time, but not information from later in the sequence. To train RNN, back-propagation through time (BPTT) is often used [51]. However, due to the gradient-vanishing and exploding problem, it is not viable to train standard RNNs with BPTT [52]. It is difficult to propagate errors from later time steps back to previous time steps enough to adjust network settings correctly. Thus, in the subsequent development of the RNNs, the gated recurrent unit (GRU) and the long short-term memory (LSTM) network have also been employed to resolve this concern [53–55].

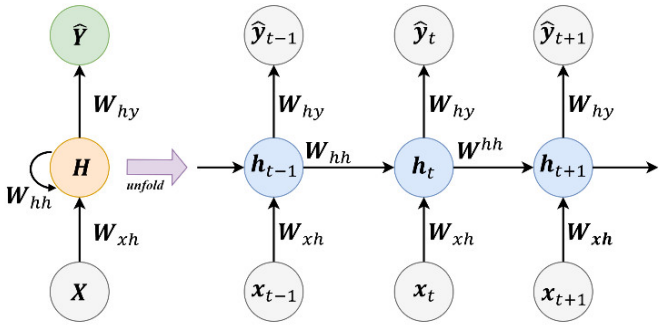


Figure 5. RNN architecture.

3.2.2. Long Short-Term Memory

Long short-term memory (LSTM) is an enhanced RNN type with memory cells [55]. These memory cells are introduced for handling long-term temporal dependencies in the data. It makes it possible to add or remove information from the current cell state. The input gate (i_t), the forget gate (f_t), and the output gate (o_t) is computed to control this memory. Thus, an important feature that appeared early in the input sequence can be propagated over a long distance by LSTM units and capture potential long-distance dependencies. Unlike RNN, LSTM has three logistic sigmoid gates and one tanh layer. The gates limit the information that passes through the cell. They determine which information is needed by the following cell and which can be rejected. The output value is typically in the range of 0 to 1, where 0 indicates "reject all" and 1 indicates "include all". The architecture of the LSTM unit is described in Figure 6, and the formulas to compute the time-step t for each hidden state in the LSTM unit are explained in the following equations [56]:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i .[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i) \tag{3}$$

$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}_f .[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f\right) \tag{4}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o .[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o) \tag{5}$$

$$\widetilde{\boldsymbol{C}}_t = \tanh(\boldsymbol{W}_c .[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_c) \tag{6}$$

$$\boldsymbol{C}_t = \boldsymbol{f}_t * \boldsymbol{C}_{t-1} + \boldsymbol{i}_t * \widetilde{\boldsymbol{C}}_t \tag{7}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t * \tanh(\boldsymbol{C}_t) \tag{8}$$

Each LSTM cell takes three inputs: $\boldsymbol{h}_{t-1}$, $\boldsymbol{C}_{t-1}$, and input $\boldsymbol{x}_t$, as well as produces two outputs: $\boldsymbol{h}_t$ and $\boldsymbol{C}_t$. $\boldsymbol{h}_t$ is the hidden state, $\boldsymbol{C}_t$ is the cell state or memory, and $\boldsymbol{x}_t$ is the current data point or input for a given time t. The first sigmoid layer (σ) contains two inputs, namely $\boldsymbol{h}_{t-1}$ and $\boldsymbol{x}_t$, where $\boldsymbol{h}_{t-1}$ is the previous cell's hidden state. It is referred to as the forget gate ($\boldsymbol{f}_t$) since its output controls the amount of the previous cell's information to be included. The output is a number in the range [0, 1] which is multiplied (point-wise) by the previous cell state $\boldsymbol{C}_{t-1}$. The second sigmoid layer is the input gate ($\boldsymbol{i}_t$), which determines what new information to add to the cell. The tanh layer then creates candidate vector $\widetilde{\boldsymbol{C}}_t$. These two layers evaluate the information to be stored in the cell state by the point-wise multiplication of ($\boldsymbol{i}_t * \widetilde{\boldsymbol{C}}_t$). The result is then added to the forget gate's prior cell state ($\boldsymbol{f}_t * \boldsymbol{C}_{t-1}$) to generate the current cell state $\boldsymbol{C}_t$. Next, the output of the cell is computed using a sigmoid and tanh layer. The sigmoid layer determines which cell state will be output, while the tanh layer adjusts the output in the range of $[-1, 1]$. Finally, a point-wise multiplication of these two layers produces the cell's output $\boldsymbol{h}_t$.

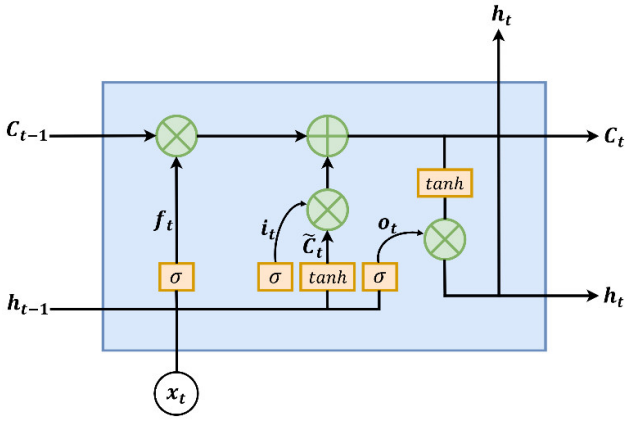


Figure 6. LSTM unit architecture.

3.2.3. Gated-Recurrent Unit

Gated Recurrent Unit (GRU) was introduced by [54] to enable each recurrent unit to adaptively capture dependencies at different time scales. It resembles LSTM but has a simpler cell design. The GRU comprises gating units that influence the flow of information inside the unit but without memory cells. GRU computes two gates known as update and reset gates that regulate the flow of information through each hidden unit. Therefore, the update and reset gates, which are depicted by colored boxes in the GRU cell in Figure 7, can be calculated using the following equations [39,57]:

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \tag{9}$$

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \tag{10}$$

$$\widetilde{h}_t = tanh(Wxhx_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \tag{11}$$

$$\widetilde{h}_t = z_t \odot h_{t-1} + (1 - z_t) \odot \widetilde{h}_t \tag{12}$$

where z_t is the update gate, r_t is the reset gate, W represents the weight vector, $\odot$ represents the element-wise multiplication, and σ is the sigmoid function. At each time t, it takes input x_t and hidden state h_{t-1} from the previous time $t-1$. Next, it outputs a new hidden state h_t for the following time t. In order to find h_t, two steps are needed in GRU. The first is generating the candidate hidden state. It multiplies the input and hidden state from the previous time $t-1$ by the reset gate output r_t. This information is then passed to the tanh function, and the result is the candidate's hidden state. In this equation, the reset gate value controls how much the previous hidden state influences the candidate state. If $r_t = 1$, all information from h_{t-1} is considered. Otherwise, if $r_t = 0$, the previous hidden state's information is ignored. Once we obtain the candidate state, it is used to generate h_t, and it is where the update gate is involved. In GRU, instead of using a separate gate like in LSTM, it utilizes a single update gate to regulate both the historical information (h_{t-1}) and the new information from the candidate state. Furthermore, z_t is critical in this equation. It can determine how much past knowledge needs to be passed into the future. The z_t value ranges from 0 to 1.

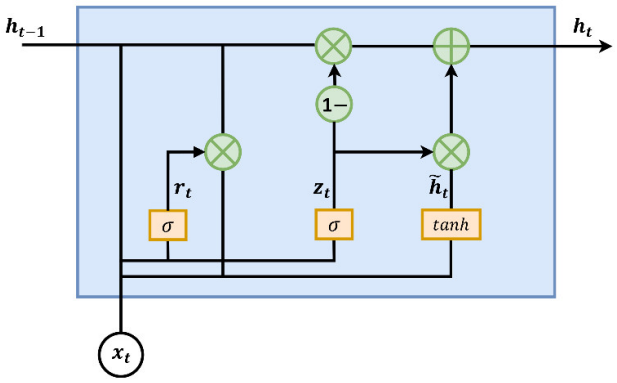


Figure 7. GRU unit architecture.

3.3. Recurrent Network-Based Ensemble Method (RNE)

The wisdom of the crowd posits that collective knowledge is better than knowledge of the few [58]. Drawing upon this, the objective of the ensemble method is to improve the prediction performance by combining the results of multiple models. An ensemble approach uses a set of model predictions from the first-level classifiers as inputs for the second-level learning model. The second-level model is trained to optimally combine the first-level classifiers' predictions to produce the final prediction. In this way, ensemble learning can turn a set of weak classifiers into a statistically strong classifier [22]. In various situations, the ensemble method has been shown to show more accurate predictions over single models. When the models return different results, the potential for performance improvement is more considerable when using an ensemble method [59].

In this paper, we designed a named-entity recognition model, namely the recurrent network-based ensemble (RNE) method to extract ingredient entities and their attributes from recipe text. RNE built based on an ensemble-learning framework consists of recurrent network models such as RNN, GRU, and LSTM. These models are trained independently on the same dataset and combined to produce better predictions in extracting food entities such as ingredient names, products, units, quantities, and states for each ingredient in a recipe. The overall architecture of this model is shown in Figure 8.

The input data to RNE is the named-entity dataset we created in Section 3.1, namely the FINER dataset. This dataset contains 11 tags in the IOB2 tagging scheme shown in Table 4. The dataset consists of 181,970 sentences and is divided into 80:20 percentage portions for training and testing. In the first layer, we retrieved the contextual meaning of words using pre-trained GloVe [60] word embeddings. In this layer, the pre-trained word embedding is used to map each word ($\boldsymbol{w}_1$, $\boldsymbol{w}_2$, $\ldots$ $\boldsymbol{w}_n$) in the sentence $\boldsymbol{X}$ to a word vector. Next, in the second layer, we train three base classifiers independently on the same dataset using the Adam optimizer and cross-entropy loss function so that the loss function converges to a better local minimum. In this study, we use three bidirectional deep learning models, namely Bi-RNN, Bi-GRU, and Bi-LSTM, as the base classifiers. Each model produces an output prediction represented as $\boldsymbol{P}_1$, $\boldsymbol{P}_2$, and $\boldsymbol{P}_3$. Next, in the combiner layer, we concatenate the three different predictions and obtain the final prediction $\boldsymbol{Y}$, as shown in Figure 9. Specifically, we draw the final predicted label $\boldsymbol{Y}$ through $\boldsymbol{Y} = \boldsymbol{\mathcal{F}}([\boldsymbol{P}_1 \circ \boldsymbol{P}_2 \circ \boldsymbol{P}_3])$, with $\circ$ denoting the concatenation operator and $\boldsymbol{\mathcal{F}} : \mathbb{R}^{(D\times 3)} \rightarrow \mathbb{R}^K$ denoting a linear layer without any non-linear activation function. Note that we indicate D as the output dimension from each recurrent model and K as the number of classes that shall be predicted by the ensemble model. In practice, we set D equal to K, and in this way we impose each recurrent model to predict the named entities while aggregating each prediction through this ensemble layer to obtain a more precise prediction. Figure 9 illustrates an output of NER sentence. All the implementation details for the experiments we carried out in this study are described in the experimental setup in the following subsection.

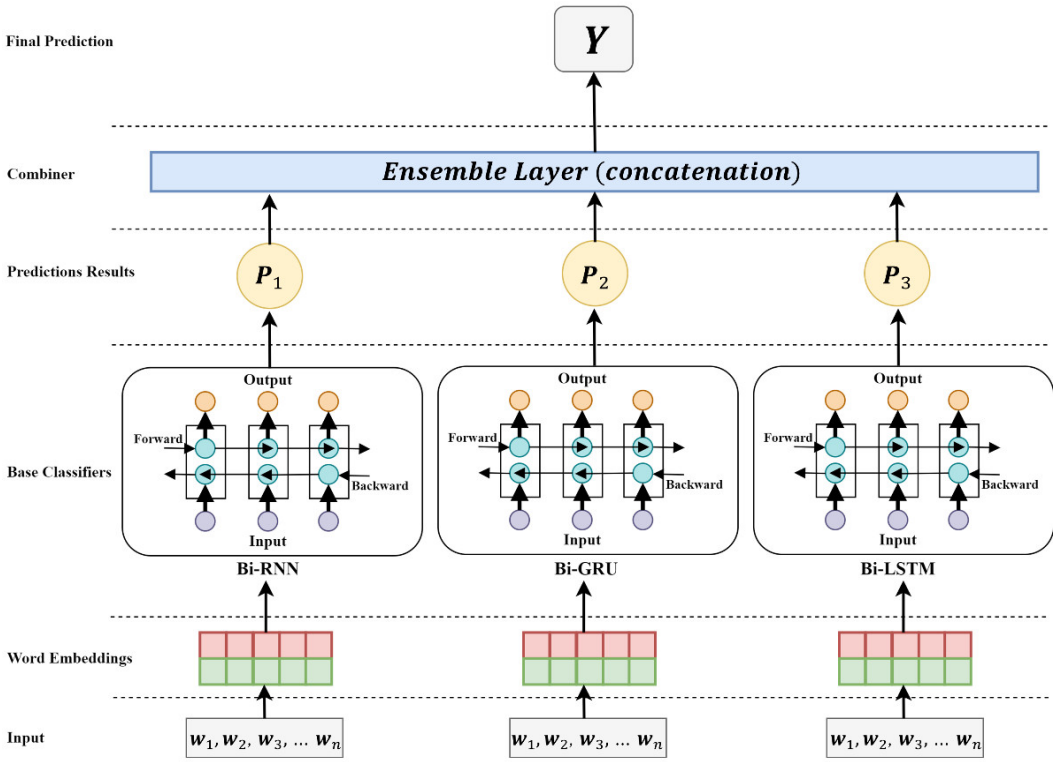


Figure 8. Organization of RNE.

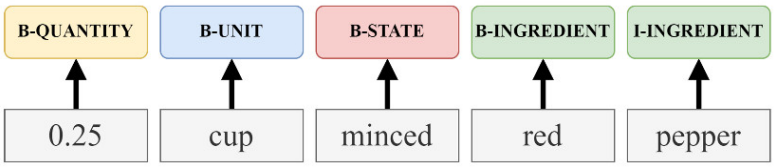


Figure 9. Output data example.

4. Experiments

This section presents the settings and results related to our experiments, including the experimental setup and the evaluation metric, and then discusses the results of the experiments performed in this study to evaluate the effectiveness of our proposed RNE method.

4.1. Experimental Setup

All of the experiments are conducted with the Pytorch python package in an open-source Anaconda GPU environment, with the 3.9 python version used in the experiment. The algorithms above are trained with the same hyperparameter settings: the GloVe word vector's dimension is 300; the hidden layer dimension's parameter is 16; the learning rate is 3×10^{-4} L2 has a regularization value of 2×10^{-5}; the optimizer is Adam; the epoch parameter is 30; and the batch parameter is 256.

4.2. Evaluation Metrics

To assess our model performance, we use three different metrics [61]: precision, recall, and F1 score. We count the true positives (TP_t), false positives (FP_t), and false negatives (FN_t). TP_t describes the number of positive samples that are correctly predicted, FP_t describes the number of positive samples that are incorrectly predicted, and FN_t describes the number of negative samples that are incorrectly predicted. In addition, we convert

the multiclass confusion matrix of each dataset into a binary confusion matrix using the one-against-all method [62]. These evaluation metrics are calculated as follows.

$$\text{Precision} = \sum_{t \in T} \frac{TP_t}{(TP_t + FP_t)} \tag{13}$$

$$\text{Recall} = \sum_{t \in T} \frac{TP_t}{(TP_t + FN_t)} \tag{14}$$

$$\text{F1} = \sum_{t \in T} \frac{2 \times Precision_t Recall_t}{(Precision_t + Recall_t)} \tag{15}$$

where:

- **Precision** is the ratio of true positives and total predicted positives;
- **Recall** is the ratio of true positives and total actual positives;
- **F1 Score** is the harmonic mean of precision and recall.

4.3. Results and Discussion

Evaluation metrics, such as precision, recall, and F1 score, are used to evaluate the proposed model. Using the proposed recurrent network-based ensemble (RNE) method model, these metrics are measured and compared with those of the individual classifiers of RNN, GRU, and LSTM. Table 6 summarizes the performance of the models on both unidirectional and bidirectional types. A one-way or unidirectional network only stores forward information (past to future), while the bidirectional network takes the input flows from both directions and utilizes information from both sides (past to future and vice versa).

Table 6. Comparative analysis of models between unidirectional and bidirectional recurrent networks. The best performance is highlighted in **bold**.

Model	F1 Score (%)	
	Unidirectional	Bidirectional
RNN	94.57	95.53
GRU	94.95	95.85
LSTM	94.98	95.70
RNE	**95.17**	**96.09**

From Table 6, we observe that the best performance came from RNE with a bidirectional network type at 96.09%, outperforming all of the single models by 0.2–0.5% in F1 score. According to the results, we conclude that the bidirectional network type is preferable to the unidirectional type. This architecture has many benefits for real-world problems, particularly in NLP. Each component of the input sequence contains information from the past and the present. Due to the combination of network layers from both directions, the bidirectional type can generate more meaningful output. On the other hand, RNN performed the worst among all models, which is expected given that GRU, LSTM, and RNE have more complex architectures than RNN. We believe that these results demonstrate that GRU and LSTM are superior to RNN due to the use of memory cells with a gate mechanism for managing the information flow in sequential data [8].

Furthermore, in Figure 10, we observed and compared the computational cost of each model. The figure shows us the average training time for four models on a computer with an Intel(R) Core (TM) i9-10900KF processor running at 3.70 GHz, an NVidia GeForce RTX 3090 graphics processing unit, and 32 GB of RAM. The Bi-LSTM model performed exceptionally faster than the other models. On average, Bi-LSTM performed 3.2% faster than Bi-RNN, the second fastest model, 7.3% faster than Bi-GRU, and 74.3% faster than RNE. The RNE model is the most time-consuming model due to the expertise and time required to train and maintain multiple models rather than a single model.

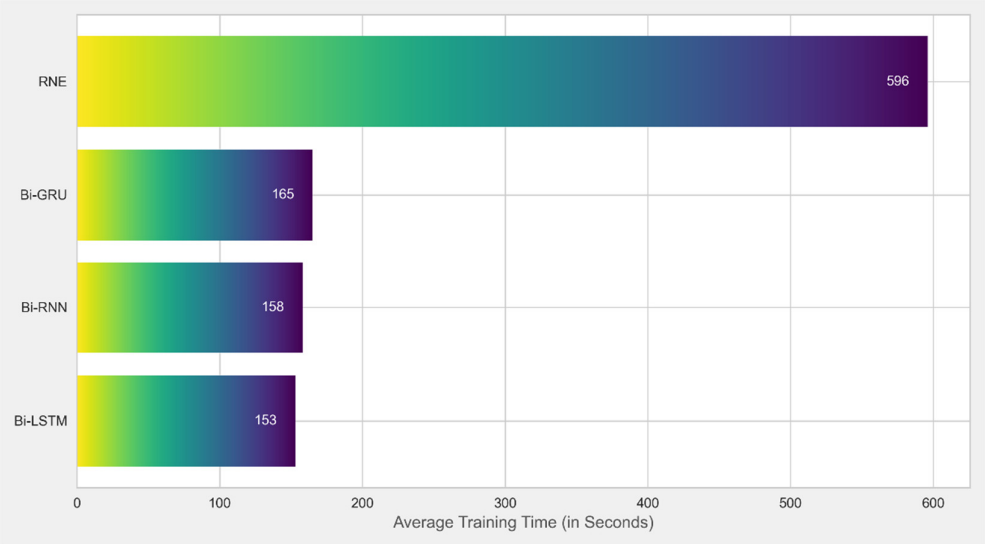


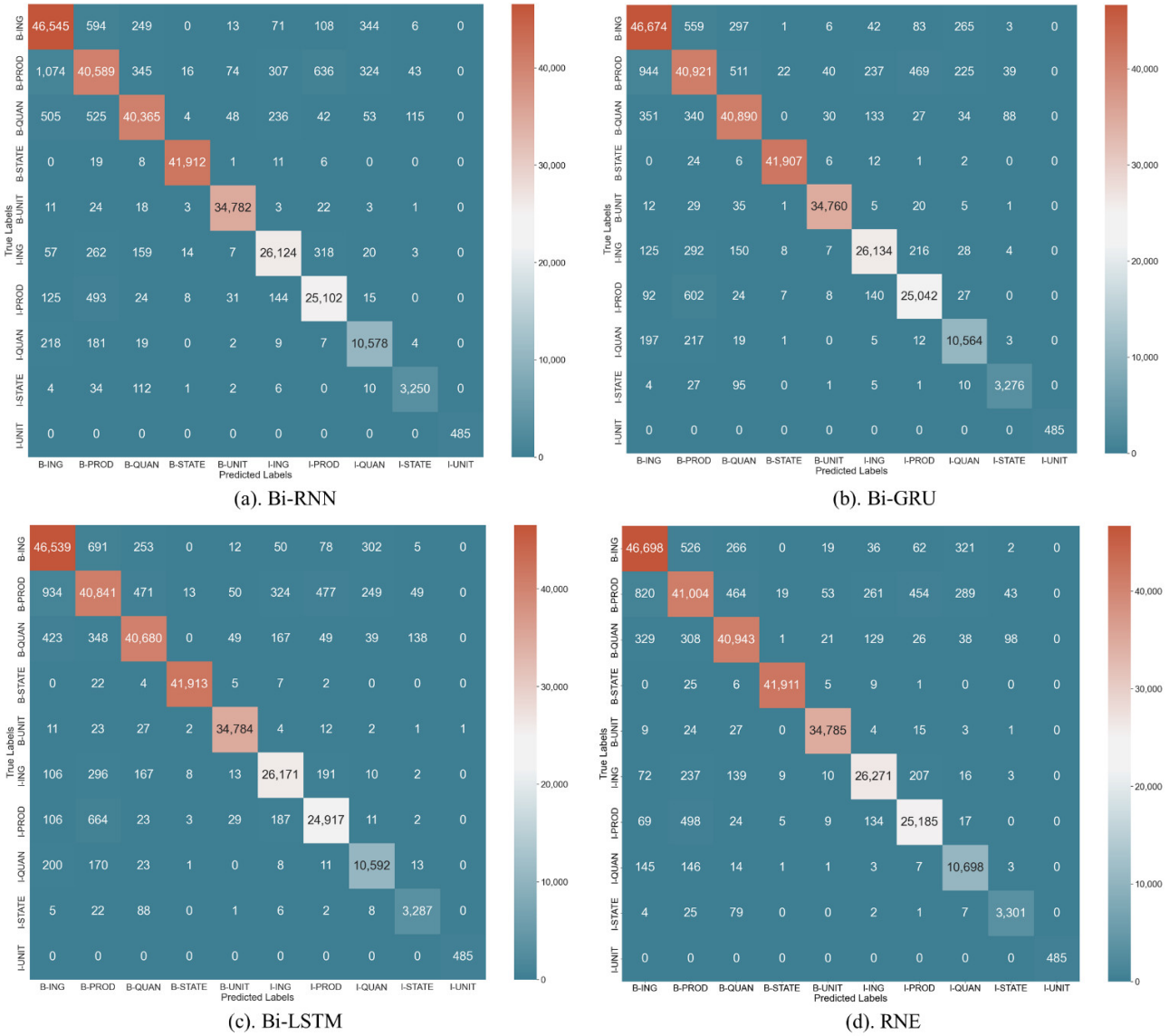
Figure 10. Computational cost comparison.

Table 7 shows detailed performance values measured for each entity for the bidirectional type of RNN, GRU, LSTM, and RNE models. Model performance ranges from 88% for B-PRODUCT and 100% for I-UNIT. For inside-tag UNIT, most models achieve perfect performance, and this is because I-UNITs are very rare (accounting for 0.17% of data in the dataset), and most of them in the training dataset belong to the "*fluid ounces*" token, making it relatively easy for the model to predict. The same is true for the performance of B-STATE, which has almost 100% accuracy in all models because even though the number of entities in the dataset is almost 9.7%, they have a small number of unique tokens, so it is easy to predict. In contrast, the B-PRODUCT entity has the lowest performance not only because it has a small distribution of annotations in the dataset (about 1.2%), but also due to the difficulty of identifying the beginning tag of the PRODUCT. The PRODUCT entity tends to be long, unique, and rare. It consists of more than one token, as shown in an example of product entity extraction in Figure 1 above. Thus, it seems that the number of training data examples for this entity is insufficient to learn an entity automatically.

The confusion matrix in Figure 11 shows that across all models, the majority of the misclassifications in the confusion matrix were caused by incorrect predictions of B-PRODUCT against all the entities, except for I-UNIT. On the contrary, I-UNIT has zero misclassifications against all the models' entities. Overall, based on the performance results shown in Tables 6 and 7, as well as on the confusion matrix in Figure 11, it is suggested that RNE can be the method of choice for NER. It can be seen that our proposed model (RNE) improves the model's performance in detecting ingredient entities by 0.2% to 0.5% compared to the single models. However, the computational cost shown in Figure 10 suggests that given the model's simplicity and computational time, a single model such as Bi-LSTM could be the second-best option after ERN. Although Bi-GRU performs marginally better, Bi-LSTMs are faster at training and computationally less expensive than ERN. In addition, it is critical to emphasize that there is no assurance that the combination of multiple classifiers will always outperform the ensemble's best individual classifier except for certain cases [63]. Consequently, combining classifiers may not necessarily outperform the best classifier in the ensemble, but it does reduce the risk of making a particularly poor selection.

Table 7. The classification report for each NER model with their best performance highlighted in **bold**. Note that **P** is precision, **R** is recall, and **F1** is F1 score.

Class	Bi-RNN			Bi-GRU			Bi-LSTM			RNE		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
B-INGREDIENT	0.948	0.971	0.960	0.954	0.974	0.964	0.956	0.971	0.964	**0.958**	**0.977**	**0.968**
B-PRODUCT	0.887	0.935	0.910	0.890	0.943	0.916	0.885	0.941	0.912	**0.891**	**0.946**	**0.917**
B-QUALITY	**0.977**	0.964	0.970	0.967	**0.976**	0.972	0.971	0.971	0.971	0.975	**0.976**	**0.975**
B-STATE	**0.999**	**0.999**	**0.999**	**0.999**	**0.999**	**0.999**	**0.999**	**0.999**	**0.999**	**0.999**	**0.999**	**0.999**
B-UNIT	0.995	**0.998**	0.996	**0.997**	0.997	**0.997**	0.995	**0.998**	**0.997**	**0.997**	0.997	**0.997**
I-INGREDIENT	0.970	0.969	0.996	0.978	0.969	0.974	0.971	0.971	0.971	**0.979**	**0.974**	**0.977**
I-PRODUCT	0.937	0.968	0.952	0.951	0.965	0.958	0.947	0.961	0.954	**0.952**	**0.970**	**0.961**
I-QUALITY	0.911	0.960	0.935	0.936	0.959	0.947	0.928	0.961	0.945	**0.937**	**0.967**	**0.951**
I-STATE	**0.943**	0.951	0.947	0.922	0.958	0.940	0.906	0.961	0.933	0.937	**0.968**	**0.953**
I-UNIT	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.998	**1.000**	0.999	**1.000**	**1.000**	**1.000**



(a). Bi-RNN (b). Bi-GRU

(c). Bi-LSTM (d). RNE

Figure 11. Confusion matrix of each model.

5. Conclusions

In this paper, a recurrent network-based ensemble method (RNE) is proposed to improve the performance of NER in extracting ingredient entities from recipe data. We combined the predicted results of bidirectional RNN, GRU, and LSTM using the ensemble concatenation approach for the final prediction. The evaluation results demonstrate that the RNE improves all the single models' performance by 0.2–0.5% of the F1 score. From these results, it can be concluded that the performance of the ensemble model is better than a single model. The increase in performance through the ensemble model occurs due to the reduction in the variance component of the prediction error made by the contributing model. Thus, it can be said that the ensemble method can improve the classifier's performance and reduce variance.

However, considering the model's simplicity and computational time, a single model like Bi-LSTM can be the second option after ERN. Although Bi-GRU performs slightly better, Bi-LSTMs are faster at training and computationally less expensive than other models. Moreover, there could be possible ways to improve these numbers and build a more robust model that maintains a balance between computational cost and model's performance by incorporating other factors, such as exploring better architectures, expanding the size of the dataset, and choosing a better hyperparameters setting. Finally, we hope that ingredient entity extraction will be a potential area of future research in many food-related systems.

Author Contributions: Conceptualization, K.S.K.; methodology, K.S.K.; software, K.S.K. and B.-K.S.; validation, B.-K.S.; data curation, K.S.K.; writing—original draft, K.S.K.; writing—review and editing, K.S.K. and B.-K.S.; visualization, K.S.K.; supervision, B.-K.S.; project administration, B.-K.S.; funding acquisition, B.-K.S. All authors have read and agreed to the published version of the manuscript.

Funding: The present study was part of the "*Future Fisheries Food Research Center Project*", funded by the Ministry of Oceans and Fisheries, Republic of Korea (grant no. 201803932).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available and can be accessed at Figshare [64]: https://doi.org/10.6084/m9.figshare.20222361 (accessed on 5 May 2022).

Acknowledgments: The authors would like to thank Taek-Jeong Nam from Future Fisheries Food Research Center for the funding support. We also thank Ahmad Wisnu Mulyadi from the Department of Brain and Cognitive Engineering, Korea University, for his advice during the revision process of this paper.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

NER	Named-Entity Recognition
NLP	Natural Language Processing
RNE	Recurrent Network-based Ensemble
FINER	Food Ingredient NER
RNN	Recurrent Neural Network
GRU	Gated-Recurrent Unit
LSTM	Long Short-Term Memory
Bi-LSTM	Bidirectional Long Short-Term Memory
BERT	Bidirectional Encoder Representations from Transformers
MLP	Multilayer Perceptron
ABM1	AdaBoostM1
ME	Maximum Entropy
CRF	Conditional Random Field
SVM	Support Vector Machine

CNN	Convolutional Neural Network
HMM	Hidden Markov Model
MEM	Maximum Entropy Model
GloVe	Global Vector
TP	True Positive
FP	False Positive
FN	False Negative

References

1. Syed, M.H.; Chung, S.-T. MenuNER: Domain-Adapted BERT Based NER Approach for a Domain with Limited Dataset and Its Application to Food Menu Domain. *Appl. Sci.* **2021**, *11*, 6007. [CrossRef]
2. Komariah, K.S.; Sin, B.-K. Nutrition-Based Food Recommendation System for Prediabetic Person. In Proceedings of the 2020 Korea Software Congress, Pyeongchang, Korea, 21–23 December 2021; pp. 660–662.
3. Kalra, J.S.; Batra, D.; Diwan, N.; Bagler, G. Nutritional Profile Estimation in Cooking Recipes. In Proceedings of the 2020 IEEE 36th International Conference on Data Engineering Workshops (ICDEW), Dallas, TX, USA, 20–24 April 2020; pp. 82–87.
4. Pellegrini, C.; Özsoy, E.; Wintergerst, M.; Groh, G. Exploiting Food Embeddings for Ingredient Substitution. In Proceedings of the HEALTHINF, Online, 11–13 February 2021.
5. Popovski, G.; Seljak, B.K.; Eftimov, T. A Survey of Named-Entity Recognition Methods for Food Information Extraction. *IEEE Access* **2020**, *8*, 31586–31594. [CrossRef]
6. Krishnan, V.; Ganapathy, V. Named Entity Recognition. Available online: http://cs229.stanford.edu/proj2005/KrishnanGanapathy-NamedEntityRecognition.pdf (accessed on 4 February 2021).
7. Wen, Y.; Fan, C.; Chen, G.; Chen, X.; Chen, M. A Survey on Named Entity Recognition. In *Communications, Signal Processing, and Systems*; Lecture Notes in Electrical Engineering; Liang, Q., Wang, W., Liu, X., Na, Z., Jia, M., Zhang, B., Eds.; Springer: Singapore, 2020; Volume 571, pp. 1803–1810, ISBN 9789811394089.
8. Li, J.; Sun, A.; Han, J.; Li, C. A Survey on Deep Learning for Named Entity Recognition. *IEEE Trans. Knowl. Data Eng.* **2022**, *34*, 50–70. [CrossRef]
9. Speck, R.; Ngonga Ngomo, A.-C. Ensemble Learning for Named Entity Recognition. In *The Semantic Web—ISWC 2014*; Lecture Notes in Computer Science; Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C., Eds.; Springer International Publishing: Cham, Switzerland, 2014; Volume 8796, pp. 519–534, ISBN 978-3-319-11963-2.
10. Ekbal, A.; Saha, S. Weighted Vote-Based Classifier Ensemble for Named Entity Recognition: A Genetic Algorithm-Based Approach. *ACM Trans. Asian Lang. Inf. Process.* **2011**, *10*, 9. [CrossRef]
11. Canale, L.; Lisena, P.; Troncy, R. A Novel Ensemble Method for Named Entity Recognition and Disambiguation Based on Neural Network. In *The Semantic Web—ISWC 2018*; Lecture Notes in Computer Science; Vrandečić, D., Bontcheva, K., Suárez-Figueroa, M.C., Presutti, V., Celino, I., Sabou, M., Kaffee, L.-A., Simperl, E., Eds.; Springer International Publishing: Cham, Switzerland, 2018; Volume 11136, pp. 91–107, ISBN 978-3-030-00670-9.
12. Wang, H.; Zhao, T.; Tan, H.; Zhang, S. Biomedical Named Entity Recognition Based on Classifiers Ensemble. *Int. J. Comput. Sci. Appl.* **2008**, *5*, 1–11.
13. Won, M.; Murrieta-Flores, P.; Martins, B. Ensemble Named Entity Recognition (NER): Evaluating NER Tools in the Identification of Place Names in Historical Corpora. *Front. Digit. Humanit.* **2018**, *5*, 2. [CrossRef]
14. Naderi, N.; Knafou, J.; Copara, J.; Ruch, P.; Teodoro, D. Ensemble of Deep Masked Language Models for Effective Named Entity Recognition in Health and Life Science Corpora. *Front. Res. Metr. Anal.* **2021**, *6*, 689803. [CrossRef]
15. Nayel, H.; Shashirekha, H.L. Improving NER for Clinical Texts by Ensemble Approach Using Segment Representations. In Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017), Kolkata, India, 18–21 December 2017; pp. 197–204.
16. Copara, J.; Naderi, N.; Knafou, J.; Ruch, P.; Teodoro, D. Named Entity Recognition in Chemical Patents Using Ensemble of Contextual Language Models. *arXiv* **2020**, arXiv:2007.12569.
17. Jiang, Z. The Application of Ensemble Learning on Named Entity Recognition for Legal Knowledgebase of Properties Involved in Criminal Cases. In Proceedings of the 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), Dalian, China, 25–27 August 2020; IEEE: Dalian, China, 2020; pp. 701–705.
18. Yadav, V.; Bethard, S. A Survey on Recent Advances in Named Entity Recognition from Deep Learning Models. In Proceedings of the 27th International Conference on Computational Linguistics, Association for Computational Linguistics, Santa Fe, NM, USA, 21–25 August 2018; pp. 2145–2158.
19. Chiu, J.P.C.; Nichols, E. Named Entity Recognition with Bidirectional LSTM-CNNs. *Trans. Assoc. Comput. Linguist.* **2016**, *4*, 357–370. [CrossRef]
20. Ma, X.; Hovy, E. End-to-End Sequence Labeling via Bi-Directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*; Long Papers; Association for Computational Linguistics: Berlin, Germany, 2016; Volume 1, pp. 1064–1074.
21. Schuster, M.; Paliwal, K.K. Bidirectional Recurrent Neural Networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673–2681. [CrossRef]

22. Aggarwal, C.C. (Ed.) *Data Classification: Algorithms and Applications*; Chapman & Hall/CRC Data Mining and Knowledge Discovery Series; Taylor & Francis Group: Boca Raton, FL, USA; CRC Press: Abingdon, UK, 2014; ISBN 978-1-4665-8674-1.
23. Popovski, G.; Kochev, S.; Seljak, B.; Eftimov, T. FoodIE: A Rule-Based Named-Entity Recognition Method for Food Information Extraction. In *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods*; SCITEPRESS—Science and Technology Publications: Prague, Czech Republic, 2019; pp. 915–922.
24. Eftimov, T.; Koroušić Seljak, B.; Korošec, P. A Rule-Based Named-Entity Recognition Method for Knowledge Extraction of Evidence-Based Dietary Recommendations. *PLoS ONE* **2017**, *12*, e0179488. [CrossRef]
25. Diwan, N.; Batra, D.; Bagler, G. A Named Entity Based Approach to Model Recipes. In Proceedings of the 2020 IEEE 36th International Conference on Data Engineering Workshops (ICDEW), Dallas, TX, USA, 20–24 April 2020; pp. 88–93.
26. Popovski, G.; Seljak, B.K.; Eftimov, T. FoodBase Corpus: A New Resource of Annotated Food Entities. *Database* **2019**, *2019*, baz121. [CrossRef]
27. Cenikj, G.; Popovski, G.; Stojanov, R.; Seljak, B.K.; Eftimov, T. BuTTER: BidirecTional LSTM for Food Named-Entity Recognition. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020; pp. 3550–3556.
28. Stojanov, R.; Popovski, G.; Cenikj, G.; Koroušić Seljak, B.; Eftimov, T. A Fine-Tuned Bidirectional Encoder Representations from Transformers Model for Food Named-Entity Recognition: Algorithm Development and Validation. *J. Med. Internet Res.* **2021**, *23*, e28229. [CrossRef] [PubMed]
29. Young, T.; Hazarika, D.; Poria, S.; Cambria, E. Recent Trends in Deep Learning Based Natural Language Processing. *CoRR* **2017**, *13*, 55–75.
30. Huang, Z.; Xu, W.; Yu, K. Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv* **2015**, arXiv:1508.01991.
31. Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; Dyer, C. Neural Architectures for Named Entity Recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*; Association for Computational Linguistics: San Diego, CA, USA, 2016; pp. 260–270.
32. Panchendrarajan, R.; Amaresan, A. Bidirectional LSTM-CRF for Named Entity Recognition. In Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation, Hong Kong, China, 1–3 December 2018; Association for Computational Linguistics: Hong Kong, China, 2018.
33. Goyal, A.; Gupta, V.; Kumar, M. Recurrent Neural Network-Based Model for Named Entity Recognition with Improved Word Embeddings. *IETE J. Res.* **2021**, 1–7. [CrossRef]
34. Gao, Y.; Wang, R.; Zhou, E. Stock Prediction Based on Optimized LSTM and GRU Models. *Sci. Program.* **2021**, *2021*, 4055281. [CrossRef]
35. Banik, N.; Rahman, M.H.H. GRU Based Named Entity Recognition System for Bangla Online Newspapers. In *Proceedings of the 2018 International Conference on Innovation in Engineering and Technology (ICIET)*; IEEE: Dhaka, Bangladesh, 2018; pp. 1–6.
36. Yan, S.; Chai, J.; Wu, L. Bidirectional GRU with Multi-Head Attention for Chinese NER. In Proceedings of the 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 12–14 June 2020; pp. 1160–1164.
37. Yang, Z.; Salakhutdinov, R.; Cohen, W.W. Multi-Task Cross-Lingual Sequence Tagging from Scratch. *arXiv* **2016**, arXiv:1603.06270.
38. Soltau, H.; Shafran, I.; Wang, M.; Shafey, L.E. RNN Transducers for Nested Named Entity Recognition with Constraints on Alignment for Long Sequences. *arXiv* **2022**, arXiv:2203.03543.
39. Chowdhury, S.; Dong, X.; Qian, L.; Li, X.; Guan, Y.; Yang, J.; Yu, Q. A Multitask Bi-Directional RNN Model for Named Entity Recognition on Chinese Electronic Medical Records. *BMC Bioinform.* **2018**, *19*, 499. [CrossRef]
40. Maclin, R.; Opitz, D.W. Popular Ensemble Methods: An Empirical Study. *arXiv* **2011**, arXiv:1106.0257.
41. Dietterich, T.G. Ensemble Methods in Machine Learning. In *Multiple Classifier Systems*; MCS 2000. Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2000; Volume 1857. [CrossRef]
42. Sarkar, D.; Natarajan, V. *Ensemble Machine Learning Cookbook: Over 35 Practical Recipes to Explore Ensemble Machine Learning Techniques Using Python*; Packt: Birmingham/Mumbai, India, 2019; ISBN 978-1-78913-660-9.
43. Zhou, Z.-H. *Ensemble Methods: Foundations and Algorithms*; Chapman & Hall/CRC Machine Learning & Pattern Recognition Series; Taylor & Francis: Boca Raton, FL, USA, 2012; ISBN 978-1-4398-3003-1.
44. Allrecipes. Available online: https://www.allrecipes.com/ (accessed on 10 September 2021).
45. SpaCy. Available online: https://spacy.io/ (accessed on 5 March 2022).
46. Loper, E.; Bird, S. NLTK: The Natural Language Toolkit. *arXiv* **2002**, arXiv:cs/0205028.
47. Doccano: Text Annotation Tool for Human. Available online: https://github.com/doccano/doccano (accessed on 5 March 2022).
48. Boushehri, S.S.; Qasim, A.B.; Waibel, D.; Schmich, F.; Marr, C. Annotation-Efficient Classification Combining Active Learning, Pre-Training and Semi-Supervised Learning for Biomedical Images. *bioRxiv* **2020**, 414235. [CrossRef]
49. Ramshaw, L.A.; Marcus, M.P. Text Chunking Using Transformation-Based Learning. In *Natural Language Processing Using Very Large Corpora*; Text Speech and Language Technology; Armstrong, S., Church, K., Isabelle, P., Manzi, S., Tzoukermann, E., Yarowsky, D., Eds.; Springer: Dordrecht, The Netherlands, 1999; Volume 11, pp. 157–176, ISBN 978-90-481-5349-7.
50. Understanding LSTM Networks—Colah's Blog. Available online: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (accessed on 3 May 2022).
51. Williams, R.J.; Peng, J. An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories. *Neural Comput.* **1990**, *2*, 490–501. [CrossRef]

52. Bengio, Y.; Simard, P.; Frasconi, P. Learning Long-Term Dependencies with Gradient Descent Is Difficult. *IEEE Trans. Neural Netw.* **1994**, *5*, 157–166. [CrossRef]
53. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv* **2014**, arXiv:1412.3555.
54. Cho, K.; van Merrienboer, B.; Bahdanau, D.; Bengio, Y. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of the SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 103–111.
55. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
56. Deng, Y.; Jiao, Y.; Lu, B.-L. Driver Sleepiness Detection Using LSTM Neural Network. In *Neural Information Processing*; Lecture Notes in Computer Science; Cheng, L., Leung, A.C.S., Ozawa, S., Eds.; Springer International Publishing: Cham, Switzerland, 2018; Volume 11304, pp. 622–633, ISBN 978-3-030-04211-0.
57. Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 1724–1734.
58. Alizadeh, H.; Yousefnezhad, M.; Bidgoli, B.M. Wisdom of Crowds Cluster Ensemble. *IDA* **2015**, *19*, 485–503. [CrossRef]
59. Oza, N.C.; Russell, S. Online Ensemble Learning. Ph.D. Thesis, University of California, Berkeley, CA, USA, 2001.
60. Pennington, J.; Socher, R.; Maning, C.D. GloVe: Global Vectors for Word Representation. Available online: https://nlp.stanford.edu/projects/glove/ (accessed on 3 May 2022).
61. Sokolova, M.; Lapalme, G. A Systematic Analysis of Performance Measures for Classification Tasks. *Inf. Process. Manag.* **2009**, *45*, 427–437. [CrossRef]
62. Allwein, E.L.; Schapire, R.E.; Singer, Y. Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. *J. Mach. Learn. Res.* **2001**, *1*, 113–141. [CrossRef]
63. Fumera, G.; Roli, F. A Theoretical and Experimental Analysis of Linear Combiners for Multiple Classifier Systems. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 942–956. [CrossRef]
64. Komariah, K.S.; Sin, B.-K.; Purnomo, A.T. FINER: Food Ingredient NER Dataset (Version 3). Figshare. 2022. Available online: https://doi.org/10.6084/m9.figshare.20222361.v3 (accessed on 5 May 2022).