informatics

MDPI

Article

SMPT: A Semi-Supervised Multi-Model Prediction Technique for Food Ingredient Named Entity Recognition (FINER) Dataset Construction

Kokoy Siti Komariah [1], Ariana Tulus Purnomo [2,3], Ardianto Satriawan [4,*], Muhammad Ogin Hasanuddin [4], Casi Setianingsih [5,*] and Bong-Kee Sin [1,6]

[1] Department of Artificial Intelligence Convergence, Pukyong National University, Busan 48513, Republic of Korea
[2] Department of Information System, Faculty of Engineering and Technology, Sampoerna University, L'Avenue Building, Jl. Raya Pasar Minggu No. Kav 16, Pancoran, Jakarta Selatan 12780, Indonesia
[3] Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology, Taipei 10607, Taiwan
[4] School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Jl. Ganesha No. 10, Bandung 40132, Indonesia
[5] School of Electrical Engineering, Telkom University, Jl. Telekomunikasi No. 1, Bandung 40257, Indonesia
[6] Division of Computer Engineering, Pukyong National University, Busan 48513, Republic of Korea
* Correspondence: asatriawan@staff.stei.itb.ac.id (A.S.); setiacasie@telkomuniversity.ac.id (C.S.)

Abstract: To pursue a healthy lifestyle, people are increasingly concerned about their food ingredients. Recently, it has become a common practice to use an online recipe to select the ingredients that match an individual's meal plan and healthy diet preference. The information from online recipes can be extracted and used to develop various food-related applications. Named entity recognition (NER) is often used to extract such information. However, the problem in building an NER system lies in the massive amount of data needed to train the classifier, especially on a specific domain, such as food. There are food NER datasets available, but they are still quite limited. Thus, we proposed an iterative self-training approach called semi-supervised multi-model prediction technique (SMPT) to construct a food ingredient NER dataset. SMPT is a deep ensemble learning model that employs the concept of self-training and uses multiple pre-trained language models in the iterative data labeling process, with a voting mechanism used as the final decision to determine the entity's label. Utilizing the SMPT, we have created a new annotated dataset of ingredient entities obtained from the Allrecipes website named FINER. Finally, this study aims to use the FINER dataset as an alternative resource to support food computing research and development.

Keywords: data creation; ingredient entity extraction; named entity recognition; self-training; pre-trained language model; ensemble voting

Citation: Komariah, K.S.; Purnomo, A.T.; Satriawan, A.; Hasanuddin, M.O.; Setianingsih, C.; Sin, B.-K. SMPT: A Semi-Supervised Multi-Model Prediction Technique for Food Ingredient Named Entity Recognition (FINER) Dataset Construction. *Informatics* **2023**, *10*, 10. https://doi.org/10.3390/informatics10010010

Academic Editor: Zhiwen Yu

Received: 15 December 2022
Revised: 9 January 2023
Accepted: 11 January 2023
Published: 13 January 2023

1. Introduction

In our daily meals, we expect to consume food and beverages with complete nutritional content, ranging from carbohydrates, proteins, vitamins, fats, and minerals. Meanwhile, many diseases, such as anemia, sprue, goiter, poor diet, and starvation, are caused by malnutrition [1]. Therefore, it is necessary to be aware of food nutrition to have an accurate ingredient profile of food [2]. This ingredient profile will also offer useful information for people who practice a specific diet or have allergies to some food ingredients [3,4]. One of the challenges for those people is obtaining food ingredients that match the recipes. Some ingredients are often difficult to find or expensive in certain areas, depending on geographic location and season. As a result, we often look for substitute ingredients with similar taste, nutrition, and texture. For people on a diet or with food allergies, information about ingredients in food recipes plays a critical role in their health. In recent years, research

enthusiasm for food computing has become a major focus of various nutrition and health research efforts [5]. One of the key technologies needed for the development is named entity recognition (NER) in food of textual recipes [6].

Named entity recognition is an information extraction technique that identifies keywords or information units dispersed within a text with known labels [7]. A named entity is a term or expression that uniquely identifies an element among a set of other elements with similar properties. It provides a piece of rough categorical information related to the target. Named entities in the text usually play key roles in a sentence both functionally and semantically. Once such food and nutrition entities are located in the text, we can further explore important information regarding the relationship between those entities. Such information can help develop intelligent applications for the food industry [3,8], such as a personalized recommendation system for individual diets, finding the ingredient substitutes for individuals with food allergies [4], and calculating nutrient levels in food to prevent malnutrition [2].

When developing reliable machine learning (ML) models, the biggest challenge comes from the need for extensive training data. Several studies on food data construction have been conducted with various approaches and data sources [5]. Table 1 compares the existing food datasets and the FINER dataset, which is the output of the proposed method. Among the datasets in Table 1, Recipe1M+ [9] and RecipeNLG [10] are used for multimodal machine learning tasks, containing both image and text.

Table 1. Comparison of available food dataset to our FINER dataset.

Dataset	Method	Source	Dataset Size (Recipes)	Entities
FoodBase [6]	Ruled-based approach	AllRecipes	1000 curated; 21,790 uncurated version	Based on Hansard corpus semantic tags: AG (food and drink) AE (animal) AF (plant)
Recipe1M+ [9]	Deep learning approach	Various cooking sites and image search engines for image data extension	1 million recipes and 13 million food images	-
RecipeDB [11]	Ruled-based approach	Food.com AllRecipes Tarladalal The Spruce Eats Epicurious Food Network Taste	118,171	Name State Unit Quantity Size Temp Dry/Fresh
RecipeNLG [10]	Deep learning approach	Recipe1M+ and auhtors private data gathered from various cooking sites	Over 1 million new data	-
TASTEset [12]	Deep learning approach	AllRecipes Food.com Tasty Yummly	700	Food Quantity Unit Process Physical Quality Color Taste Purpose Part
FINER (Ours)	Deep learning approach	AllRecipes	64,782	Ingredient Product Quantity Unit State

However, our study focuses on named entity recognition in food recipe text. There are datasets available for food entity recognition, but they are usually few regarding the

availability of training samples [13]. The issue of limited data on food and its attributes has recently been addressed in several papers with proposals for the construction of food NER datasets [6,11]. Batra et al. presented RecipeDB [11] while Popovski et al. discussed the FoodBase corpus, which was developed using a rule-based approach [6]. While publicly available, it is rather small, with only 1000 recipes for the curated (manually evaluated) and 21,790 recipes for the uncurated version. Similar to FoodBase, a recently published dataset called TASTEset [12] has relatively limited data, consisting of 700 recipes. On the other hand, the RecipeDB dataset, built by Batra et al., [11], is relatively large and covers a wide range of recipes since it is derived from numerous online food recipe-sharing sites. However, we cannot directly use the dataset because it is intended for search applications about recipes or food ingredients.

Our study focuses on extracting entities in the recipe texts from the Allrecipes website, a popular online social media for sharing food recipes. We propose a novel iterative framework called the semi-supervised multi-model prediction technique (SMPT) to construct a new dataset of annotated ingredient entities from recipes text called food ingredients named entity recognition (FINER) using ML techniques to address the data limitation issue. SMPT employs the self-training idea that builds on pre-trained models in the iterative data labeling process [14]. After a predetermined number of iterations, the resulting dataset is re-appended to the training set, which quickly expands to the desired size. Self-training is a semi-supervised learning strategy that repeatedly trains a base classifier and augments the training set with newly machine-labeled instances [15,16]. Our proposed SMPT is an iterative algorithm that alternates the following steps until every unlabeled sample is assigned a label:

1. Train a base classifier on annotated data.
2. Use this classifier to predict labels for unlabeled data and move some of the confident samples into the labeled set.

Along with self-training, SMPT also uses of pre-trained models such as spaCy NER [17], BERT [18], and DistilBERT [19], which help improve the label quality. Then finally, a voting scheme is used to determine the entity's label.

This study aims to develop a high-quality NER dataset for food and ingredients with the following main contributions:

1. The SMPT method. It is a deep ensemble learning model that adopts the self-training concept that builds on pre-trained language model (LM) in the iterative data labeling process. Then, the voting scheme is used as the final decision to determine the entity's label. Furthermore, this approach can be applied to other domains besides food and nutrients.
2. The FINER dataset. It is an annotated dataset for food ingredient entities. The dataset is made public and accessible on Figshare [20].

The remaining sections are organized as follows: Section 2 discusses previous works on data construction and their approaches. Section 3 explains the proposed method, such as the data construction workflow, data preparation, NER labeling format, and our machine learning approach for annotating the dataset. Then, Section 4 describes the experimental results with discussion. Finally, Section 5 concludes the paper.

2. Related Works

Due to recent advancements in food computing [5], the importance of digital text data in the food domain has just recently gained more attention. It has brought a new dimension to food information processing. One technology behind many applications and solutions rests on NER as a keystone for natural language processing (NLP) in text. Vast research has been conducted to develop NER [21–23], which include rule-based, ML-based, and hybrid approaches. This section will provide a summary of previous efforts to create food information processing, including the dataset construction method, transfer learning, the pre-trained LM, and self-training in NER research development.

A rule-based approach was developed by Popovski et al. [6] to create a standard silver dataset in food NER named FoodBase corpus. This relatively new corpus was compiled from the Allrecipes website. They performed statistical evaluation using mean, median, mode, and standard deviation to evaluate the dataset and differentiate it into curated and uncurated versions. In more recent research, Popovski et al. [13] evaluated four distinct NER techniques using the FoodBase corpus: FoodIE, NCBO (SNOMED CT), NCBO (OntoFood), and NCBO (FoodON). In addition, Cenikj et al. proposed BuTTER [24] to enhance the FoodBase corpus' performance. BuTTER is a NER approach that employs bidirectional long short-term memory (Bi-LSTM) and conditional random fields (CRFs) to extract food entities from 1000 annotated recipes.

Similar research in creating the food NER dataset was conducted by Batra et al. [11]. They employed a rule-based approach and *k*-means clustering to build the RecipeDB database. In building RecipeDB, they extracted the data from several food sites, such as Allrecipes [25], Food.com [26], Tarladalal [27], The Spruce Eats [28], Epicurious [29], Food Network [30], and Taste [31]. Hence, using the newly created RecipeDB dataset, Diwan et al. [32] performed NER on the dataset using the Stanford NER model with *k*-means clustering to group the vectors of ingredients. The groups are formed based on part-of-speech (POS) tagging through the bag-of-words approach. The decision to select the number of formed clusters is based on the group formation's weakness and the group's interpretation.

Transfer learning, pre-trained LMs, and self-training have recently become a trend in NLP research development. Transfer learning enables the rapid development of accurate models without the need to start the learning process from scratch. We can utilize the patterns learned by a particular model when solving a different problem. In the case of NER dataset creation, Kim et al. [33] proposed a method to automatically annotate the unlabeled data with a bagging-based bootstrapping approach using CRF and transfer learning. The resulting dataset was then verified by two versions of deep neural network (DNN)-based NER models: Bi-LSTM-CRF and vanilla BERT. Our proposed SMPT employs an iterative process similar to that of Kim et al. However, instead of utilizing a bootstrapping strategy with CRF to annotate the unlabeled data, we implemented self-training and applied it to pre-trained models during the data-labeling procedure.

Self-training is a part of a semi-supervised learning strategy that repeatedly trains a base classifier and augments the training set with newly machine-labeled instances [15,16]. It is often used because of its advantages in overcoming the limitations of small labeled data. SentAugment is one such effort proposed by Du et al. [34] to address the issue of insufficient unlabeled data for self-training in the language domain. It uses sentence embedding to extract unlabeled data in the sample domain from a large corpus. It also performs self-training on the retrieved sentences. SentAugment is a data augmentation technique that employs a pre-trained model similar to our prior study in [35], but in SentAugment, they combined it with self-training.

Additionally, pre-trained LM are frequently exploited to develop NLP applications due to their advantages. One of the commonly used pre-trained LM in NLP is BERT. BERT uses the encoders of the transformer architecture [18]. It has several advantages, such as being easy to integrate across various use cases and tasks, achieving high performance quickly without excessive fine-tuning and avoiding the requirement for massive labeled datasets for training [36]. In particular, Stojanov et al. [37] introduced FoodNER as a set of corpus-based food NER. They used BERT-based pre-trained models—BERT and BioBERT (both standard and large)—to extract the dataset in the FoodBase corpus. Fine-tuning the three pre-trained BERT models on five semantic resource groups yielded fifteen different models in FoodNER. Another study by Anna et al. [12] proposed a dataset called TASTEset. They manually annotated 700 recipe texts and then trained them on two transformer-based models, BERT and LUKE. Their best model achieved over 90% F1-score on average.

On the other hand, Pellegrini et al. [4] proposed two models for ingredient embedding: word embedding (Food2Vec) and BERT-based (FoodBERT). These models were used to

identify ingredient substitutes for cooking recipes. The results of the Recipe1M+ dataset showed that FoodBERT is better than the BERT model regarding food knowledge.

Moreover, the current data creation trends indicate that combining semi-supervised learning and pre-training can offer additional benefits [14–16]. Therefore, we attempt to leverage the benefit of combining the two methods in this research to address the issue of limited datasets for entity recognition tasks, especially in the food domain.

3. Dataset Construction

3.1. Dataset Construction Workflow

The data construction workflow is comprised of four stages: (i) data preparation; (ii) manual data labeling; (iii) training and automatic data labeling; and (iv) final data evaluation. Throughout the construction, we utilize several NLP libraries such as spaCy [17], NLTK [38], and Doccano [39] annotation tools. Figure 1 illustrates the detailed data construction.

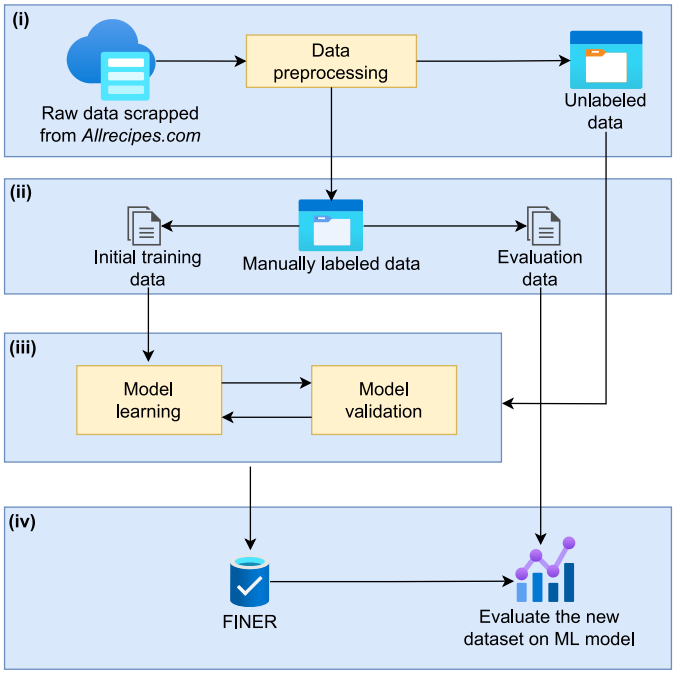


Figure 1. FINER dataset construction workflow.

A detailed explanation of each stage is given in the following:

(i) **Data preparation.** We begin by cleaning the text data collected from the Allrecipes website, followed by a number of pre-processing steps. The explanation is provided in detail in Section 3.2.

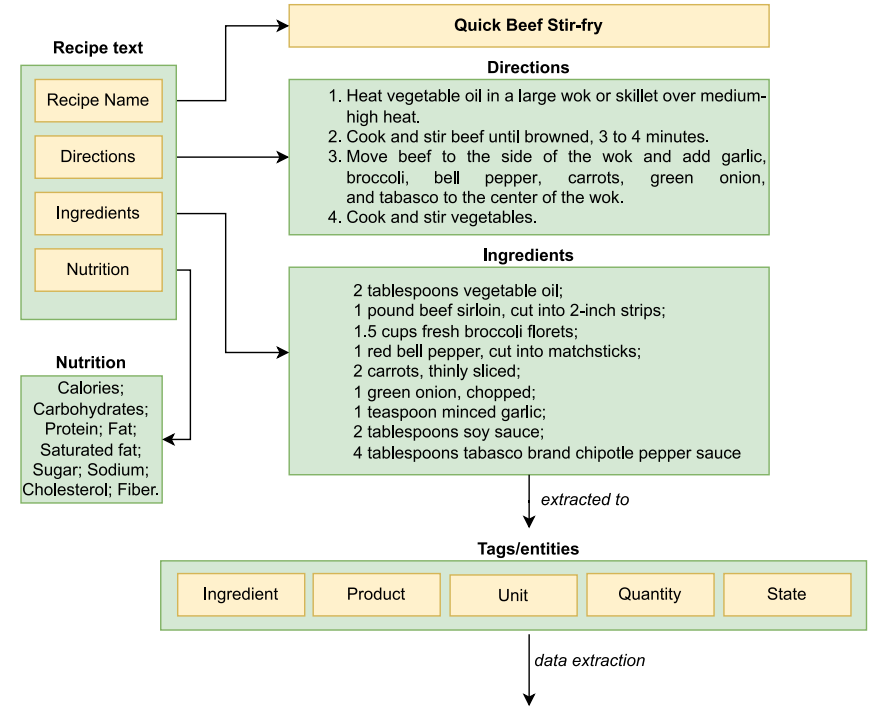
(ii) **Manual data labeling.** We manually label 2000 instances and split them in half for the initial training and evaluation sets. Using the initial training set, a baseline NER annotator is developed. The evaluation set is preserved for the final evaluation stage of the complete dataset.

(iii) **Training and automatic data labeling.** In this stage, a baseline model is developed utilizing the initial training set from the previous stage. This model is then applied to a set of unlabeled data to predict its labels. Then, we have a newly created set of labeled data, some of which have been incorporated into the previous set. This procedure is repeated until no more unlabeled data are available.

(iv) **Final data evaluation.** After a number of repetitions, we reside at the dataset named Food Ingredient NER or FINER. Using several classifiers, including CRF, Bi-LSTM, and BERT, we indirectly evaluate the quality of the dataset. Their performance is evaluated using the reserved evaluation set.

3.2. Data Preparation

The dataset in this study is based on the recipe text scraped from the Allrecipes website [25]. The recipe text consists of recipe names, ingredients, direction, and nutrition, as depicted in Figure 2. This study extracts ingredient entities and their properties, such as quantity, unit, and ingredient status, from the ingredient section. It contains 64,782 raw recipes containing eight food categories: breakfast and brunch, dinner, main dishes, side dishes, drinks, dessert, bread, and salad. The data have been cleaned up so that the data format matches the input requirements. Then, it is followed by the data pre-processing. Data pre-processing involves multiple steps such as: tokenizing the data, transforming it into lower-case, eliminating special characters, stop words, punctuation symbols, white spaces, and then lemmatizing the data.



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 sauce		tabasco brand chipotle sauce		4	

Figure 2. Ingredient data extraction from Allrecipes dataset.

We then define additional rules as follows:

1. Each phrase in the ingredients section is split into individual phrases to simplify the extraction procedure. Then after preprocessing, the resulting dataset consists of 181,970 phrases.

2. Standardize the unit and quantity measurements. For example, in the units, all abbreviations are converted to their true form; thus, "tbsp" becomes "tablespoon". In quantity, we convert all fractional numbers to decimal form so that "½" becomes "0.5".
3. Since our dataset comprises a list of ingredients, stop words and punctuation may not always be meaningless to the text's intent, but they could help interpret entities. Therefore, we have created custom lists of stop words and punctuation. For example, *"1 (2 ounces) package butter"* indicates that one package of butter equals 2 ounces. Although *"2 ounces"* is enclosed in parentheses, we keep the parentheses since they provide information for translating the number of ingredients to standard units.

After a series of pre-processing steps, the recipe text data are divided into three sets, as summarized in Table 2. For training and evaluation data, we manually annotated the first two sets using Doccano annotation tools [39] and left the remaining set unlabeled, which will be labeled recursively by the proposed method of the NER annotator.

Table 2. The organization of the initial dataset. Initial training data and evaluation data are manually annotated.

Dataset	Total (# of Sentences)
Initial Training Data	1000
Evaluation Data	1000
Unlabeled Data	179,970
Total	181,970

3.3. Named Entity Labeling

Entities usually represent an important chunk of a particular sentence. Named entity recognition is a technique to detect and classify atomic elements in a text into predefined categories or classes that vary depending on the domain of interest. People commonly classify them into names of persons, organizations, events, dates, and many more in the general domain. The objective of this study is to extract food ingredient entities and their properties from recipe texts. In this study, we define five entity classes, each associated with an entity tag explained as follows:

1. **INGREDIENT:** the name of the food or ingredient. For example, garlic, apple, carrots, vegetable oil, etc.
2. **PRODUCT:** the food or ingredient from a specific brand mentioned. Examples include Archer farms dark chocolate hot cocoa mix, Tabasco brand chipotle pepper sauce, etc.
3. **QUANTITY:** The amount of the food or ingredient associated with the unit. Examples: 1½, 25, 0.5, etc.
4. **STATE:** The processing state of the food or ingredient. For example, chopped, grilled, minced, cut into 2-inch strips, etc.
5. **UNIT:** a measurement unit, such as pound, gram, fluid ounce, tablespoon, cup, etc.

We used the IOB2 format to chunk the entity word [7]. The main distinction between the IOB and IOB2 formats is that the IOB2 format includes the "B-tag" at the beginning of each chunk (all chunks begin with a "B-tag"). The IOB format is described in Table 3. Tags are prefixed by "B", "I", or "O" to indicate their position within the entity. The tag "O" indicates that a token is not a chunk. For example, in the ingredients entity at the beginning of a chunk, the "tag" is substituted with a named entity label, such as "B-INGREDIENT".

Table 3. The IOB tagging format.

Tag	Description
B (Begin)	Denotes that the tag is the start of a chunk.
I (Inside)	denotes that the tag is located inside a chunk.
O (Outside)	Identifies a token as non-chunk (outside).

3.4. Semi-Supervised Multi-Model Prediction Technique (SMPT)

Figure 3 depicts a general procedure of our proposed method. Our SMPT approach is an iterative step similar to the bootstrapping method used by Kim et al. in [33], but instead of using CRF for bootstrapping and automatically labeling the unlabeled data. We adopt the concept of self-training in the data labeling process. Given the small labeled datasets, we train a baseline classifier based on pre-trained models of spaCy, BERT, or DistilBERT and use them to increase the labeled set to the final selection of the token (entity) classes made by majority voting. The resulting labeled dataset is incorporated into the input for the next iteration. The whole procedure is repeated until the unlabeled sample is labeled. Thus, we present our proposed method for building the ingredient-named entity dataset in the following sections.

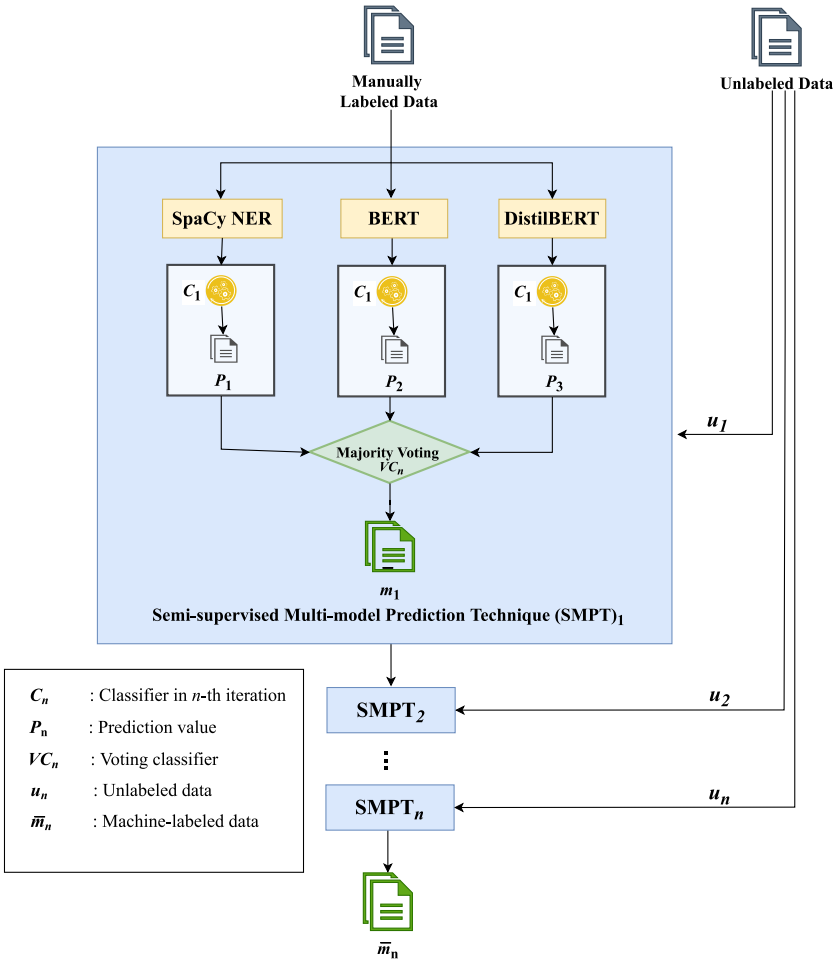


Figure 3. SMPT method for ingredient entity data extraction.

3.4.1. Models

- **spaCy NER [17]**: spaCy is a Python and Cython-based open-source library for natural language processing that provides various NLP tools for tokenization, POS-tagging, and named entity recognition text.

Figure 4 shows the procedure of spaCy being used to train a custom NER model. It employs word embedding and a multilayer CNN with residual connections. It supports pre-trained models in multiple languages and provides a default classifier for a wide range of named or numerical entities, including person, location, organization, date, and event. In addition, it allows us to extend the NER model with new classes for novel entities.

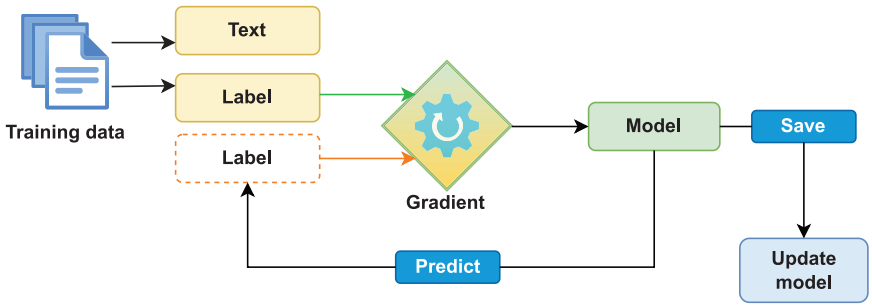


Figure 4. NER custom model training of spaCy.

In this study, we design a neural network with a custom vector layer initialized with the pre-trained spaCy's output layer. That layer is then trained using the spaCy library pre-training command [17,40] on a domain-specific text corpus. Later in the experiment, we also use the pre-trained spaCy model (*en_core_web_lg*) and train our custom dynamic embedding model on our ingredient dataset. We apply this domain-specific word embedding model to vectorize tokens while conducting transfer learning from the spaCy pre-trained model over the annotated data.

- **BERT**: BERT [18] is a language representation model that uses stacked transformer encoders that learn deep bidirectional representation from a large unlabeled corpus. An additional output layer is added to fine-tune the representation in downstream NLP tasks. Fine-tuning slightly modifies the neural network architecture for improved predictions in target tasks while training the whole network. Pre-trained BERT inherits the model weights learned during the pre-training, allowing downstream tasks to benefit from these powerful representations rather than learning from scratch.

The Transformer is equipped with multi-head attention that concatenates h different attention layers with different initializations [18,41,42]. The multi-head attention function can be calculated as follows:

$$\text{Multihead}(Q, K, V) = \text{Concat}(head_1, .., head_h)W^O \tag{1}$$

where, $head_i$ is i^{th} attention head which is given by:

$$head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2}$$

which is computed using these projection matrix parameters $W_i^K \in \mathbb{R}^{d_{emb} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{emb} \times d_V}$, $W_i^Q \in \mathbb{R}^{d_{emb} \times d_k}$, and $W^O \in \mathbb{R}^{hd_v \times d_{emb}}$ where Q, K, and V are input matrices for query, key, and values, respectively. The input matrix X is used for all three matrices at the beginning. Then their projections XW_i^Q, XW_i^K, and XW_i^V become Q_i, K_i, and V_i. These matrices are used to calculate the scaled dot-product attention for each head as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} V\right) \tag{3}$$

For the initialization of the ingredient recognition task, we use the weights of a pre-trained BERT model (*bert-base-uncased*) as shown in Figure 5. The BERT architecture is kept, but the input and output are adapted to our task.

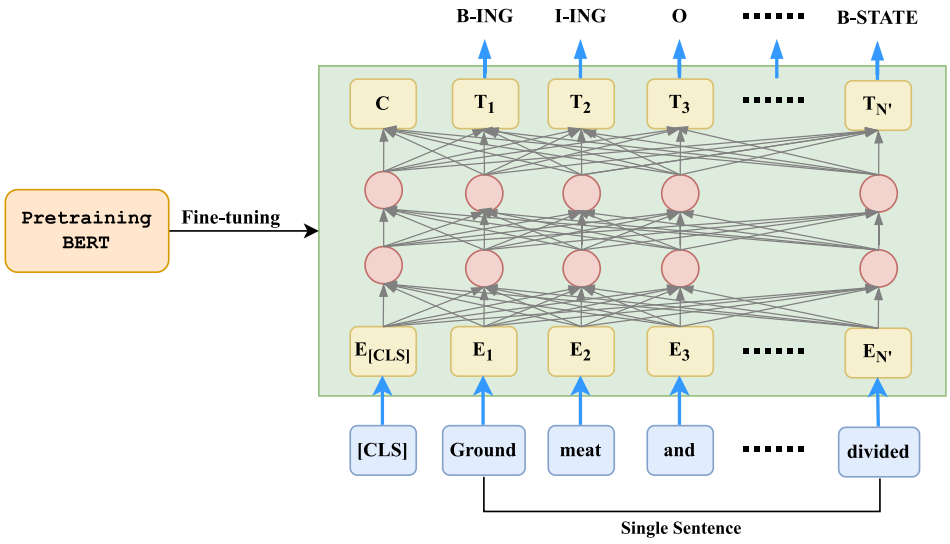


Figure 5. Fine-tuned BERT for FINER.

- **DistilBERT**: DistilBERT [19] is a compact version of BERT and is claimed to be lighter and faster than BERT with roughly comparable performance. It has 40% lesser parameters compared to *bert-base-uncased* and performs 60% faster with over 95% of the performance of BERT, as evaluated on the GLUE language understanding benchmark in this study [19]. To reduce the computational requirements of modern large neural networks, DistilBERT uses a knowledge distillation technique known as teacher-student learning. *Knowledge distillation* is a compression technique that entails training a small model to replicate the behavior of a larger model. As shown in Figure 6, the masked language model (MLM) loss is used to train the student model and the cross-entropy loss between the teacher and the student. This mechanism encourages the student model to generate a probability distribution over the predicted tokens as close to that of the teachers as possible.

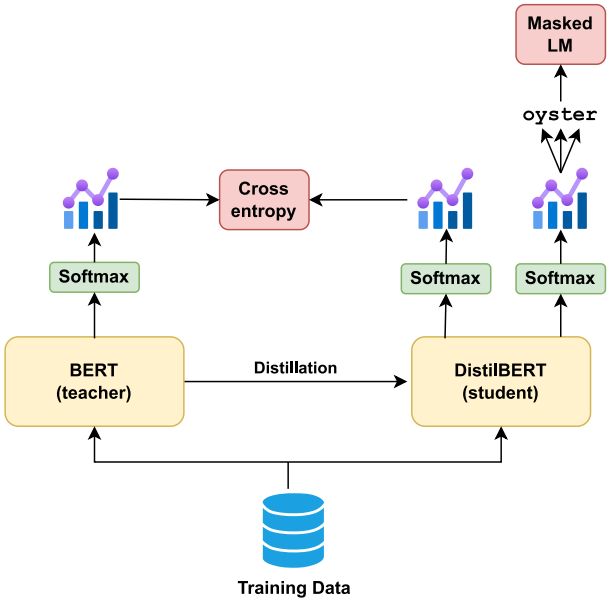


Figure 6. Knowledge distillation from BERT with the combination of cross entropy and the masked LM objectives.

In teacher–student training, a student network is trained to replicate the distribution of a teacher network's complete output or its knowledge. This procedure helps to generalize the student model in the same way as the teacher. We use cross-entropy over the soft targets (probability of the teacher) to transfer knowledge from the teacher to the student rather than training with a cross-entropy over hard targets (one-hot encoding of the gold class). Thus, the training loss becomes

$$L = -\sum_i t_i \log(s_i), \tag{4}$$

where t represents the logits of the teacher and s represents the student's logits. This loss provides a richer training signal since a single example imposes significantly more constraints than a single hard target. To further expose the mass of the distribution over the classes, Hinton et al. introduce a softmax temperature [43].

$$pi = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}. \tag{5}$$

When T tends to 0, the distribution becomes a Kronecker (equivalent to the one-hot target vector), whereas it becomes a uniform distribution when $T \longrightarrow +\infty$. During training, the same temperature parameter is applied to both the student and the teacher, revealing more signals for each training example. In inference, T is set to 1, and the standard Softmax is recovered. Therefore, using teacher signals allows us to train a smaller LM later called DistilBERT. In the implementation, we utilize the pre-trained DistilBERT model (*distilbert-base-uncased*) in the same way that we implement the BERT model.

3.4.2. Training and Labeling Methods

The key component of the proposed method is training the classifiers that will be developed in this subsection. The SMPT training process comprises two main components: the training process and the dataset labeling scheme with the dataset growth factor. Therefore, we will describe these two components in the following parts.

A. Training

The core tasks of classifier training and data labeling in SMPT consist of three steps:

1. In the first step, we develop a set of C baseline classifiers using our initial training set (1000 manually annotated instances). In our experiment, the classifiers include spaCy NER, BERT, and DistilBERT.
2. In the second step, each classifier C makes its own predictions for the test set. However, the final decisions on the unlabeled tokens are made by the majority voting scheme:

$$\bar{m} = \arg\max_{m \leq M} \sum_{c=1}^{C} d_{c,m} \tag{6}$$

   where $\bar{m}$ is the final prediction label (class), M is number of classes, $C = \mid C \mid$ the number of classifiers, and $d_{c,m}$ denotes the vote given to class m by classifier c. If the max vote is not unique, the token will be given "O" label representing that the token is not a chunk.
3. Finally, the above machine-labeled tokens with unanimous votes are considered reliable and promoted into the training set of labeled instances for the next generation of classifiers. These procedures were repeated until all tokens were labeled.

B. Dataset Building Scheme

Following the iteration procedure described in the previous section, we build an NER dataset of labeled tokens. In each iteration, we add up a fixed amount of newly labeled samples to the current training set. We define s as the growth factor. Here, each time, the

amount of addition is set to be s times that of the current training set t where $s > 0$. The number of employed schemes may vary. However, in this implementation, we decide to use the following three schemes:

- Scheme 1: $s = 2$
- Scheme 2: $s = 5$
- Scheme 3: $s = 10$

In each scheme, the labeled set grows at a different pace. Table 4 explain the data size growth procedure for data training and labeling using scheme 1 with growth factor 2 as an example. According to Table 4, we have an initial training set that we prepared in Section 3.2 with the size of t_1 and use it to create a set of baseline models or voting classifiers VC. Then we start the iteration process using VC_1. We assign labels to the unlabeled data from which we pick up $u_1 = s \times t_1$ for a promotion to the training set of size $t_2 = t_1 + s \times t_1$ for the next round. Therefore, the size of the unlabeled data that we will annotate will constantly increase by two times the size of the training data used to train the models. The procedure is repeated until all data are assigned and labeled.

Table 4. The size of the sets and additions over iterations. t_n is the training set size, u_n is the size of increments, s is growth factor, while $n = 1, 2, \ldots, n$.

Iteration	The Set	Scheme 1 (s = 2)
1	$t_1 = 1000$	$u_1 = 2 * t_1$
2	$t_2 = t_1 + 2t_1$	$u_2 = 2 * t_2$
3	$t_3 = t_2 + 2t_2$	$u_3 = 2 * t_3$
4	$t_4 = t_3 + 2t_3$	$u_4 = 2 * t_4$
.	.	.
n	$t_n = t_{n-1} + 2t_{n-1}$	$u_n = s * t_n$

In the following experiments, we explore the three schemes and compare the quality of machine-labeled data in term of the performance and computation time of tester models.

3.5. Evaluation

The built dataset is then used for the tasks that require entity recognition. Therefore, it is natural to evaluate it in similar contexts. We first build classifiers for NER and evaluate their performance with an indirect measure of the dataset quality. We compute three different metrics as follows [44,45]:

1. **Recall** is the fraction of correctly predicted positive samples (TP) in their classes:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{7}$$

2. **Precision** is the proportion of correctly predicted positive samples among the total positive predictions:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{8}$$

3. **F1-score** is a metric that measures the model's accuracy on a dataset, defined as the harmonic mean of precision and recall:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

where TP, FP, TN, and FN are:

- **TP (True Positive)**, occurs when the outputs of the NER for input tokens exactly match the same ingredient entity in the ground truth dataset.
- **FP (False Positive)**, falsely predicted positive occurs when something that is not an ingredient entity is classified as being one.

- **TN (True Negative)**, the correct negative prediction occurs when the NER method correctly predicts that the token is not an ingredient entity in the ground truth dataset.
- **FN (False Negative)**, occurs when a specific annotation is omitted when the entity should be classified as an ingredient entity. It happens when the ingredient entity is not properly extracted using the NER method.

4. Experimental Results and Data Analysis

In this section, we construct a dataset for NER in food recipes based on data drawn from Allrecipes [25]. We gather 64,782 recipes and divide them into 181,970 sentences. Then we apply the proposed SMTP method, carrying out the incremental process of NER extraction to obtain the FINER dataset. The detailed specification of the dataset is shown in Table 5, and the distribution of the dataset over entities is presented in Table 6. The FINER dataset consists of 1,397,960 words in total. Among them, 220,300 words are out of chunk entities (*O* tag), and the rest of the 1,177,660 words are ingredient entities divided into ten classes. The objective of the experiments here is to verify the effectiveness of the SMTP method in generating the FINER dataset and assess the dataset's quality using the existing NER models. In addition, our dataset of this study is available in [20].

Table 5. The detailed specification of dataset.

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 tags (without O tag)	10

Table 6. The dataset distribution for named entities.

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	2.353	0.17
O	220,300	15.76
Total	**1,397,960**	**100**

4.1. Test Results with Training Scheme

The SMPT grows the labeled dataset by multiplying the set by a certain factor. This paper considers three typical growth schemes to show the effect of the amount of training data with respect to data quality and process efficiency. Figure 7 gives a detailed picture of the performance of the three schemes with factors 2, 5, and 10. Panels (a), (b), and (c) show that the annotators' performances increase as the dataset grows over iterations. These results match our expectation that the models improve with more training data, which strongly indicates the dataset quality. The last panel (d) compares the three schemes in terms of the best F-1 scores for each.

Table 7 shows the computation time spent in each iteration to label the data for each scheme. When training time is included, scheme 1 is the slowest due to the increased number of iterations, which is five. Scheme 2 is faster but has the lowest performance among the three schemes, according to Figure 7d. On the other hand, scheme 3 takes a similar amount of time but with fewer iterations and shows superior performance to the other two. Hence, scheme 3 is the preferred method for time efficiency and performance.

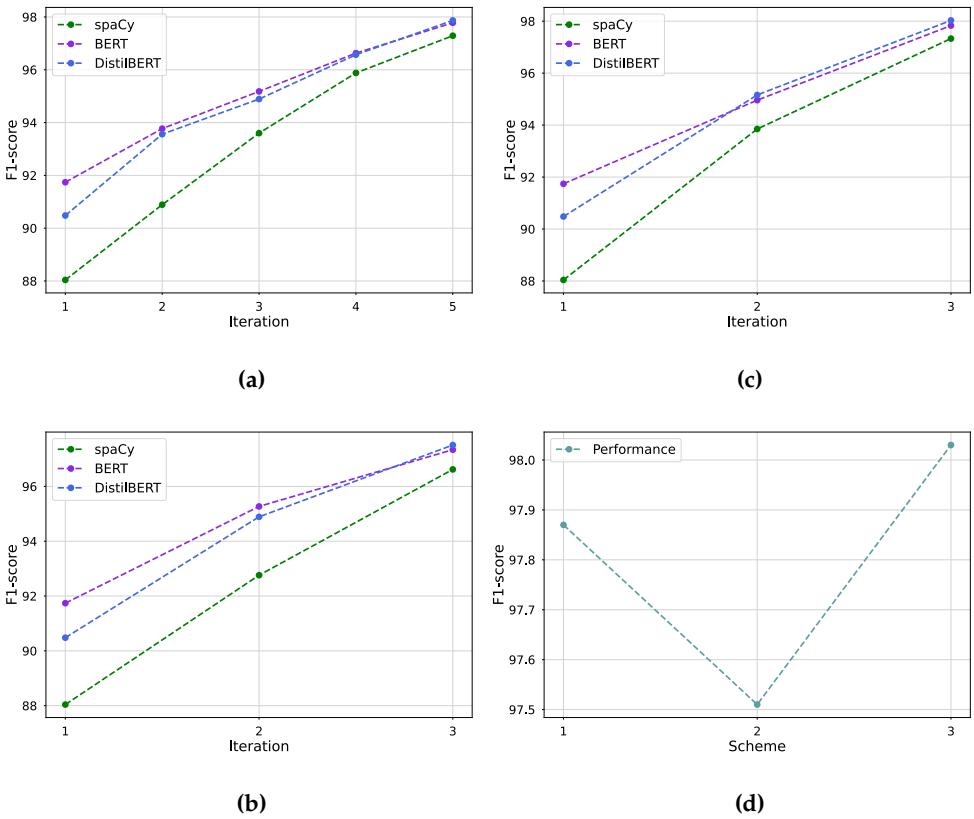


Figure 7. The performance results from 3 different schemes. Each scheme has a different total number of data processing for each iteration: (**a**) scheme 1, $s = 2$; (**b**) scheme 2, $s = 5$; (**c**) scheme 3 $s = 10$; and (**d**) best performance in each scheme.

Table 7. Computation time and their aspect of the proposed method. As the training set grows, the labeling time increases. Note that the three schemes increase the labeled set differently.

Iteration	Scheme 1 ($s = 2$)		Scheme 2 ($s = 5$)		Scheme 3 ($s = 10$)	
	Data	Time (second)	Data	Time (second)	Data	Time (second)
1	2000	146	5000	268	10,000	914
2	6000	392	30,000	1.757	110,000	10.134
3	18,000	1.365	144,970	12.099	59,970	5.018
4	54,000	4.916	-	-	-	-
5	99,970	9.436	-	-	-	-
Total	**179,970**	**16.255**	**179,970**	**14.124**	**179,970**	**16.066**

Meanwhile, as for the scenario of the sample generated by Scheme 3 over three iterations, see Figure 8 in particular. The sample input was the sentence: "*1 pound mixed domestic and wild mushrooms such as shiitake oyster or cremini, trimmed and quartered, salt and freshly ground pepper*". In the first iteration, all three models over-generated entities, some with wrong labels as highlighted in the red squares, and the correct prediction is highlighted in the blue squares. For instance, the model returned several "O" tags for non-entity words. Then in the second iteration, the model began to learn. However, in the spaCy NER model, one error is still found in classifying "*shiitake oyster*", which is an "INGREDIENT", as a "PRODUCT" class. However, in the last iteration, all models managed to detect all entities correctly in the sentence.

SpaCy NER	BERT	DistilBERT
[('1', 'QUANTITY'), ('pound', 'UNIT'), ('mixed domestic', 'INGREDIENT'), ('and', 'O'), ('wild', 'INGREDIENT'), ('mushrooms', 'INGREDIENT'), ('such', 'O'), ('as', 'O'), ('shiitake oyster', 'PRODUCT'), ('or', 'O'), ('cremini', 'INGREDIENT'), ('trimmed and quartered', 'STATE'), (',', 'O'), ('salt', 'INGREDIENT'), ('and', 'O'), ('freshly', 'STATE'), ('ground pepper', 'INGREDIENT')]	[('1', 'QUANTITY'), ('pound', 'UNIT'), ('mixed', 'O'), ('domestic', 'O'), ('and', 'O'), ('wild mushrooms', 'INGREDIENT'), ('such', 'O'), ('as', 'O'), ('shiitake oyster', 'INGREDIENT'), ('or', 'O'), ('cremini', 'INGREDIENT'), ('trimmed and quartered', 'STATE'), (',', 'O'), ('salt', 'INGREDIENT'), ('and', 'O'), ('freshly', 'STATE'), ('ground pepper', 'INGREDIENT')]	[('1', 'QUANTITY'), ('pound', 'UNIT'), ('mixed domestic and wild mushrooms', 'INGREDIENT'), ('such', 'O'), ('as', 'O'), ('shiitake oyster', 'INGREDIENT'), ('or', 'O'), ('cremini', 'INGREDIENT'), ('trimmed and quartered', 'STATE'), (',', 'O'), ('salt', 'INGREDIENT'), ('and', 'O'), ('freshly', 'STATE'), ('ground pepper', 'INGREDIENT')]

(a). Iteration 1

SpaCy NER	BERT	DistilBERT
[('1', 'QUANTITY'), ('pound', 'UNIT'), ('mixed domestic and wild mushrooms', 'INGREDIENT'), ('shiitake oyster', 'PRODUCT'), ('cremini', 'INGREDIENT'), ('trimmed and quartered', 'STATE'), ('salt', 'INGREDIENT'), ('freshly', 'STATE'), ('ground pepper', 'INGREDIENT')]	[('1', 'QUANTITY'), ('pound', 'UNIT'), ('mixed domestic and wild mushrooms', 'INGREDIENT'), ('shiitake oyster', 'INGREDIENT'), ('cremini', 'INGREDIENT'), ('trimmed and quartered', 'STATE'), ('salt', 'INGREDIENT'), ('freshly', 'STATE'), ('ground pepper', 'INGREDIENT')]	[('1', 'QUANTITY'), ('pound', 'UNIT'), ('mixed domestic and wild mushrooms', 'INGREDIENT'), ('shiitake oyster', 'INGREDIENT'), ('cremini', 'INGREDIENT'), ('trimmed and quartered', 'STATE'), ('salt', 'INGREDIENT'), ('freshly', 'STATE'), ('ground pepper', 'INGREDIENT')]

(b). Iteration 2

SpaCy NER	BERT	DistilBERT
[('1', 'QUANTITY'), ('pound', 'UNIT'), ('mixed domestic and wild mushrooms', 'INGREDIENT'), ('shiitake oyster', 'INGREDIENT'), ('cremini', 'INGREDIENT'), ('trimmed and quartered', 'STATE'), ('salt', 'INGREDIENT'), ('freshly', 'STATE'), ('ground pepper', 'INGREDIENT')]	[('1', 'QUANTITY'), ('pound', 'UNIT'), ('mixed domestic and wild mushrooms', 'INGREDIENT'), ('shiitake oyster', 'INGREDIENT'), ('cremini', 'INGREDIENT'), ('trimmed and quartered', 'STATE'), ('salt', 'INGREDIENT'), ('freshly', 'STATE'), ('ground pepper', 'INGREDIENT')]	[('1', 'QUANTITY'), ('pound', 'UNIT'), ('mixed domestic and wild mushrooms', 'INGREDIENT'), ('shiitake oyster', 'INGREDIENT'), ('cremini', 'INGREDIENT'), ('trimmed and quartered', 'STATE'), ('salt', 'INGREDIENT'), ('freshly', 'STATE'), ('ground pepper', 'INGREDIENT')]

(c). Iteration 3

Figure 8. An NER sample generated in scheme 3. Blue and red squares indicate correct and incorrect labeling, respectively.

4.2. Evaluation on Machine Learning Models

To demonstrate the quality of the FINER dataset generated by the SMPT, we indirectly evaluate this dataset using three popular NER models to infer the dataset's quality. Those models are CRF [46–48], BiLSTM-CRF [24,49–53], and BERT [3,19,33]. They have proven effective for token classification tasks, such as NER [21–23].

The three models were trained on our FINER dataset, and we utilized the reserved evaluation set of 1000 annotated samples, which we prepared in Section 3.2. Table 8 summarizes the performance of the models on the evaluation dataset in terms of precision, recall, and F1 score. BERT achieved the best performance in both micro and macro averages. The BERT micro-average yields precision, recall, and F1-score of 0.978, 0.980, and 0.979, respectively. Its macro-average metrics are slightly lower than the micro-averages. We attribute this to the imbalanced data among classes where only a few inside tags are observed for some classes in the training set. The macro-average takes the mean of the score of classes, whereas the micro-average takes the class proportion into account. As a consequence, the poor performance of a small class has a disproportionate impact on the overall performance when using the macro versions.

Table 8. Performance of NER models. The best performance is highlighted in bold.

	CRF		BiLSTM-CRF		BERT	
	micro-avg	**macro-avg**	**micro-avg**	**macro-avg**	**micro-avg**	**macro-avg**
Precision	0.953	0.950	0.973	0.956	**0.978**	**0.961**
Recall	0.964	0.957	0.974	0.962	**0.980**	**0.971**
F1-score	0.958	0.953	0.973	0.959	**0.979**	**0.966**

Additionally, the CRF model used in this experiment achieved the scores listed in Tables 8 and 9 with an appropriate hyperparameter tuning as the CRF contained a number of parameters that need fine-tuning to improve performance. However, when compared to BiLSTM-CRF and BERT, the CRF performed the poorest, which is expected given that BiLSTM-CRF has a more complex architecture than CRF and has been demonstrated to be superior due to the use of a bidirectional (forward and backward) LSTM for learning the hidden text representation and a CRF for tag decoding. Along with the FINER evaluation of the three models, we compare the results of our study with similar studies, such as RecipeDB and TASTEset. We choose RecipeDB and TASTEset for our comparison because of the similar entities and data source description to our FINER dataset.

Table 9. The classification report for each models and the best performance is emphasized in bold.

Class	**CRF**			**BiLSTM-CRF**			**BERT**		
	Precision	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
B-INGREDIENT	0.948	0.951	0.949	0.969	0.974	0.972	**0.979**	**0.981**	**0.980**
B-PRODUCT	0.909	0.896	0.902	0.932	0.959	0.946	**0.963**	**0.972**	**0.967**
B-QUANTITY	0.998	0.998	0.998	0.999	**0.999**	**0.999**	**1.000**	**0.999**	**0.999**
B-STATE	0.955	0.947	0.951	0.969	0.971	0.970	**0.981**	**0.979**	**0.980**
B-UNIT	0.994	0.994	0.994	0.996	0.997	0.997	**0.999**	**0.998**	**0.998**
I-INGREDIENT	0.929	0.956	0.942	0.958	0.975	0.967	**0.979**	**0.976**	**0.977**
I-PRODUCT	0.846	0.923	0.883	0.918	0.973	0.945	**0.927**	**0.986**	**0.956**
I-QUANTITY	**0.992**	0.958	0.975	0.982	0.985	0.989	**0.992**	**0.994**	**0.993**
I-STATE	0.929	0.951	0.940	0.952	0.978	0.965	**0.964**	**0.983**	**0.974**
I-UNIT	**1.000**	**1.000**	**1.000**	0.999	0.998	0.999	**1.000**	**1.000**	**1.000**

Compared to them, the FINER dataset outperformed both performances by an F1 score of 97.9%, as shown in Table 10.

Table 10. Comparative analysis of the FINER dataset with other similar datasets from previous work with the best performance is emphasized in bold.

Dataset	**Model**	**Performance (F1 Score)**
RecipeDB [11,32]	*K*-Means Clustering	0.961
TASTEset [12]	BERT	0.935
FINER (Ours)	BERT	**0.979**

For further analysis and verification, we present a detailed evaluation of each entity tag of these models in Table 9. Classes such as UNIT and QUANTITY were recognized with very high performance for both beginning (B) and inside (I) chunks due to their distinct morphological characteristics and the inclusion of numerical characters. Moreover, UNIT has a high possibility of appearing shortly after QUANTITY, making their prediction relatively straightforward. Overall, BERT performed better than other models in most cases. The BERT was pre-trained on a massive corpus, and it preserves powerful representations of the language, so it is not surprising that it performs best in most downstream tasks, such

as NER. Based on these experiments, we conclude that our method is helpful and beneficial in building a reasonably good dataset for typical NER tasks.

5. Conclusions

This study proposed a semi-supervised method for building a NER dataset iteratively by incorporating the concept of self-training applied to pre-trained models, such as spaCy NER, BERT, and DistilBERT, under a voting scheme mechanism for improved prediction. We call this proposed method the semi-supervised multi-model prediction technique or SMPT. In the set of experiments, a new dataset named FINER was constructed for food entity recognition and tested to verify its quality. The FINER dataset of this study is presented as a public dataset for named entity recognition in the food domain and can be accessed at Figshare [20].

The construction of FINER aims to address a small dataset with limited information, such as food and its attributes. Thus, it facilitates the future development of an integrated information system about food. To investigate the proposed method and its generated dataset performance, first, we tried and compared three schemes in annotating the unlabeled dataset iteratively, and later we evaluated the quality of our FINER dataset using three different NER models, namely CRF, BiLSTM-CRF, and BERT. They all returned significantly good performance results with precision, recall, and F1-score of over 90%. Furthermore, we compared FINER with other similar datasets from previous works, and it turned out that FINER outperformed them with the highest F1 score. These evaluation results suggest that the proposed method can help to build a useful dataset with reasonable quality.

Author Contributions: This work was realized through the collaboration of all authors. Conceptualization, K.S.K., A.T.P. and B.-K.S.; methodology, K.S.K., A.T.P. and B.-K.S.; investigation, K.S.K. and B.-K.S.; resources, B.-K.S.; data curation, K.S.K.; writing—original draft preparation, K.S.K.; writing—review and editing, A.S.; visualization, A.S.; supervision, B.-K.S.; project administration, B.-K.S.; funding acquisition, B.-K.S., A.S., M.O.H. and C.S. All authors have read and agreed to the published version of the manuscript.

Funding: The 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). The article processing charge was funded by the School of Electrical Engineering and Informatics, Institut Teknologi Bandung; the School of Electrical Engineering, Telkom University; and the Faculty of Engineering and Technology, Sampoerna University.

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 [20]: https://doi.org/10.6084/m9.figshare.20222361.v3 (accessed on 7 April 2022).

Acknowledgments: The authors would like to thank Taek-Jeong Nam from Future Fisheries Food Research Center for the project 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

The following abbreviations are used in this manuscript:

BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional Long Short-Term Memory
CNN	Convolutional Neural Network
CRF	Conditional Random Fields
FINER	Food Ingredient Named Entity Recognition

LM	Language Model
LSTM	Long Short-Term Memory
ML	Machine Learning
MLM	Masked Language Model
NER	Named Entity Recognition
NLP	Natural Language Processing
SMPT	Semi-Supervised Multi-Model Prediction Technique

References

1. Saunders, J.; Smith, T. Malnutrition: Causes and consequences. *Clin. Med.* **2010**, *10*, 624–627. [CrossRef]
2. Kalra, J.; 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.
3. 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]
4. Pellegrini, C.; Ozsoy, E.; Wintergerst, M.; Groh, G. Exploiting Food Embeddings for Ingredient Substitution. In Proceedings of the 14th International Joint Conference on Biomedical Engineering Systems and Technologies—HEALTHINF, Vienna, Austria, 11–13 February 2021; SciTePress: Setúbal, Portugal, 2021; pp. 67–77.
5. Min, W.; Jiang, S.; Liu, L.; Rui, Y.; Jain, R. A survey on food computing. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 92. [CrossRef]
6. Popovski, G.; Seljak, B.K.; Eftimov, T. FoodBase corpus: A new resource of annotated food entities. *Database* **2019**, *2019*. [CrossRef] [PubMed]
7. Krishnan, V.; Ganapathy, V. Named Entity Recognition. Stanford Lecture CS229. 2005. Available online: http://cs229.stanford.edu/2005/KrishnanGanapathy-NamedEntityRecognition.pdf (accessed on 4 February 2021).
8. Komariah, K.S.; Shin, B.K. Nutrition-Based Food Recommendation System for Prediabetic Person. In Proceedings of the Korea Software Congress 2020 (KSC 2020), Seoul, Republic of Korea, 21–23 December 2020; pp. 660–662.
9. Marin, J.; Biswas, A.; Ofli, F.; Hynes, N.; Salvador, A.; Aytar, Y.; Weber, I.; Torralba, A. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *43*, 187–203. [CrossRef] [PubMed]
10. Bień, M.; Gilski, M.; Maciejewska, M.; Taisner, W.; Wisniewski, D.; Lawrynowicz, A. RecipeNLG: A cooking recipes dataset for semi-structured text generation. In Proceedings of the 13th International Conference on Natural Language Generation, Dublin, Ireland, 15–18 December 2020; pp. 22–28.
11. Batra, D.; Diwan, N.; Upadhyay, U.; Kalra, J.S.; Sharma, T.; Sharma, A.K.; Khanna, D.; Marwah, J.S.; Kalathil, S.; Singh, N.; et al. Recipedb: A resource for exploring recipes. *Database* **2020**, *2020*, baaa077. [CrossRef]
12. Wróblewska, A.; Kaliska, A.; Pawłowski, M.; Wiśniewski, D.; Sosnowski, W.; Ławrynowicz, A. TASTEset–Recipe Dataset and Food Entities Recognition Benchmark. *arXiv* **2022**, arXiv:2204.07775.
13. 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]
14. Boushehri, S.S.; Qasim, A.B.; Waibel, D.; Schmich, F.; Marr, C. Systematic comparison of incomplete-supervision approaches for biomedical imaging classification. *bioRxiv* **2021**. [CrossRef]
15. Zoph, B.; Ghiasi, G.; Lin, T.Y.; Cui, Y.; Liu, H.; Cubuk, E.D.; Le, Q. Rethinking pre-training and self-training. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 3833–3845.
16. Lee, H.W.; Kim, N.R.; Lee, J.H. Deep neural network self-training based on unsupervised learning and dropout. *Int. J. Fuzzy Log. Intell. Syst.* **2017**, *17*, 1–9. [CrossRef]
17. spaCy. Available online: https://spacy.io/ (accessed on 5 March 2022).
18. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
19. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv* **2019**, arXiv:1910.01108.
20. Komariah, K.S.; Purnomo, A.T.; Sin, B.K. FINER: Food Ingredient NER Dataset. Available online: https://doi.org/10.6084/m9.figshare.20222361.v3 (accessed on 7 April 2022).
21. Yadav, V.; Bethard, S. A survey on recent advances in named entity recognition from deep learning models. *arXiv* **2019**, arXiv:1910.11470.
22. Li, J.; Sun, A.; Han, J.; Li, C. A survey on deep learning for named entity recognition. *IEEE Trans. Knowl. Data Eng.* **2020**, *34*, 50–70. [CrossRef]
23. Goyal, A.; Gupta, V.; Kumar, M. Recent named entity recognition and classification techniques: A systematic review. *Comput. Sci. Rev.* **2018**, *29*, 21–43. [CrossRef]
24. 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.
25. Allrecipes. Available online: https://www.allrecipes.com/ (accessed on 14 December 2022).
26. Food. Available online: https://www.food.com/ (accessed on 14 December 2022).

27. Tarla Dalal Indian Recipes. Available online: https://www.tarladalal.com/ (accessed on 14 December 2022).
28. The Spruce Eats. Available online: https://www.thespruceeats.com/ (accessed on 14 December 2022).
29. Epicuriuous. Available online: https://www.epicurious.com/ (accessed on 14 December 2022).
30. Food Network. Available online: https://www.foodnetwork.com/ (accessed on 14 December 2022).
31. Taste. Available online: https://www.taste.com.au/ (accessed on 14 December 2022).
32. 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.
33. Kim, J.; Ko, Y.; Seo, J. Construction of machine-labeled data for improving named entity recognition by transfer learning. *IEEE Access* **2020**, *8*, 59684–59693. [CrossRef]
34. Du, J.; Grave, E.; Gunel, B.; Chaudhary, V.; Celebi, O.; Auli, M.; Stoyanov, V.; Conneau, A. Self-training improves pre-training for natural language understanding. *arXiv* **2020**, arXiv:2010.02194.
35. Komariah, K.S.; Sin, B.K. BERT Pre-trained Models for Data Augmentation in Twitter Medical Named-Entity Recognition. In Proceedings of the Korea Computer Congress 2021 (KCC 2021), Jeju, Republic of Korea, 23–25 June 2021; pp. 870–872.
36. Arslan, Y.; Allix, K.; Veiber, L.; Lothritz, C.; Bissyandé, T.F.; Klein, J.; Goujon, A. A comparison of pre-trained language models for multi-class text classification in the financial domain. In *Companion Proceedings of the Web Conference 2021 (WWW '21)*; Association for Computing Machinery: New York, NY, USA, 2021; pp. 260–268. [CrossRef]
37. Stojanov, R.; Popovski, G.; Cenikj, G.; Seljak, B.K.; 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]
38. Bird, S.; Klein, E.; Loper, E. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2009.
39. Nakayama, H.; Kubo, T.; Kamura, J.; Taniguchi, Y.; Liang, X. doccano: Text Annotation Tool for Human. 2018. Available online: https://github.com/doccano/doccano (accessed on 11 February 2022).
40. Partalidou, E.; Spyromitros-Xioufis, E.; Doropoulos, S.; Vologiannidis, S.; Diamantaras, K. Design and implementation of an open source Greek POS Tagger and Entity Recognizer using spaCy. In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, Thessaloniki, Greece, 14–17 October 2019; pp. 337–341.
41. Thickstun, J. *The Transformer Model in Equations*; University of Washington: Seattle, WA, USA, 2021.
42. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *arXiv* **2017**, arXiv:1706.03762.
43. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, *2*, arXiv:1503.02531.
44. Zheng, A. *Evaluating Machine Learning Models: A Beginner's Guide to Key Concepts and Pitfalls*; O'Reilly Media: Sebastopol, CA, USA, 2015.
45. Grandini, M.; Bagli, E.; Visani, G. Metrics for multi-class classification: An overview. *arXiv* **2020**, arXiv:2008.05756.
46. Sutton, C.; McCallum, A. An introduction to conditional random fields. *Found. Trends Mach. Learn.* **2012**, *4*, 267–373. [CrossRef]
47. Patil, N.; Patil, A.; Pawar, B. Named entity recognition using conditional random fields. *Procedia Comput. Sci.* **2020**, *167*, 1181–1188. [CrossRef]
48. Komariah, K.S.; Shin, B.K. Medical Entity Recognition in Twitter using Conditional Random Fields. In Proceedings of the 2021 International Conference on Electronics, Information, and Communication (ICEIC), Jeju, Republic of Korea, 31 January–3 February 2021; pp. 1–4.
49. Lee, C. LSTM-CRF models for named entity recognition. *IEICE Trans. Inf. Syst.* **2017**, *100*, 882–887. [CrossRef]
50. Chiu, J.P.; Nichols, E. Named entity recognition with bidirectional LSTM-CNNs. *Trans. Assoc. Comput. Linguist.* **2016**, *4*, 357–370. [CrossRef]
51. 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.
52. Ma, X.; Hovy, E. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv* **2016**, arXiv:1603.01354.
53. Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; Dyer, C. Neural architectures for named entity recognition. *arXiv* **2016**, arXiv:1603.01360.