Thesis for Degree of Doctor of Philosophy

Deep Ensemble Methods for Food Ingredient Entity Recognition in Natural Language Processing

by

Kokoy Siti Komariah

Department of Artificial Intelligence Convergence

The Graduate School

Pukyong National University

February, 2023

Deep Ensemble Methods for Food Ingredient Entity Recognition in Natural Language Processing
자연어 처리에서 식품 성분 용어 인식을 위한 심층 앙상블 방법

Advisor: Prof. Sin Bong-Kee

by

Kokoy Siti Komariah

A thesis submitted in partial fulfillment of requirements
for degree of

Doctor of Philosophy

Department of Artificial Intelligence Convergence
The Graduate School
Pukyong National University

February, 2023

Deep Ensemble Methods for Food Ingredient Entity Recognition in Natural Language Processing

A dissertation

by

Kokoy Siti Komariah

Approved by:

Professor Song Ha Joo,
(Chairman)

Professor Kim Chang Soo,
(Member)

Professor Kong Kyeongbo,
(Member)

Professor Kim Hoon-Hee,
(Member)

Professor Sin Bong-Kee,
(Member)

February 17th, 2023

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
NLP	Natural Language Processing
NER	Named-Entity Recognition
FINER	Food Ingredient NER
RNE	Recurrent Network-based Ensemble
SMPT	Semi-supervised Multi-model Prediction Technique
TL	Transfer Learning
LM	Language Model
RNN	Recurrent Neural Network
GRU	Gated-Recurrent Unit
LSTM	Long-Short Term Memory
BPPT	Back Propagation Through Time
Bi-LSTM	Bidirectional Long-Short Term Memory
BERT	Bidirectional Encoder Representations from Transformers
MLM	Masked Language Modelling
DistilBERT	Distillation BERT
ELMo	Embeddings from Language Models
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
Word2vec	Word to vector representations
TP	True Positive
FP	False Positive
FN	False Negative
CoNLL	Conference on Computational Natural Language Learning
NLTK	Natural Language Toolkit

자연어 처리에서 식품 성분 용어 인식을 위한 심층 앙상블 방법

Kokoy Siti Komariah

부 경 대 학 교 대 학 원 인공지능융합학과

요 약

최근 몇 년 동안 요리를 배우거나 메뉴를 기획하려는 사람들 사이에서 레시피 공유 사이트가 많은 인기를 얻고 있다. 사용자들은 이 레시피를 이용해 자신의 라이프스타일과 건강 상태에 맞는 재료를 선택할 수 있고, 온라인으로 올라온 레시피의 정보는 다양한 음식, 영양, 그리고 건강 관리 응용 프로그램을 만드는 등 다양하게 응용될 수 있다. 그러나 온라인으로 공유되는 레시피는 구조화된 정보가 부족하다. 이 정보들을 잘 구성된 데이터로 추출하기 위해 명명된 엔티티 인식 또는 NER 이라는 자연어 처리 기술을 사용할 수 있다. 이 기술은 텍스트에서 핵심정보 또는 엔티티를 식별하고 이를 미리 정해진 범주로 분류하는 기술이다. 그러나 식품 영역에서 NER 을 개발할 때 세 가지 주요 문제가 생기는데, 첫번째로 식품 영역에 대한 데이터 세트의 가용성은 현재까지도 제한적인 것, 두번째로 식품 개체를 인식하는 데 효과적이고 효율적인 기계 학습 모델을 설계하는 것에 대한 어려움, 세번째로 기존의 NER 모델은 단일 모델에만 의존하고 있는데, NER 에 대해 앙상블 학습을 사용하는 연구가 거의 없으며, 특히 식품 영역에 대한 연구는 없다고 볼 수 있는 것이다.

이 연구에서는 다양한 학습 알고리즘을 결합하여 기존 모델의 성능을 넘어서는 집단적인 성능을 얻는 앙상블 학습 기법을 통해 이러한 문제를 해결하고자 한다.

앙상블 기법을 활용하여 위에서 언급한 문제에 대한 해결책을 다음과 같이 제안한다. 첫 번째로, SMPT(준지도 다중 모델 예측 기법)라는 반복적인 자체 훈련 접근법을 구축했다. SMPT 는 자체 훈련 개념을 채택하고 반복 데이터 레이블링 프로세스에서 사전 훈련된 여러 언어 모델을 기반으로 하는 딥 앙상블 학습 모델로, 엔티티 레이블을 결정하는 최종 결정으로 투표 메커니즘이 사용된다. 이 SMPT 를 활용하여 FINER 데이터 세트라는 새로운 주석이 달린 성분 엔티티 데이터 세트를 만들었다. 두 번째로, 이 연구에서는 Recurrent Network-based Ensemble Model (RNE)이라고 불리는 식품 성분 NER 모델을 제안한다. RNE 는 RNN, GRU 및 LSTM 을 포함한 반복 네트워크 모델과 심층 앙상블 학습을 통합하여 식품 관련 엔티티를 추출하기 위한 새로운 모델이다. 실험 결과를 통해서, 제안된 RNE 모델이 단일 모델보다 식품 조리법에서 정보를 더 효과적으로 추출할 수 있음을 보여주고 있고, 향후 추가적으로 생성되는 정보는 수많은 식품 관련 정보 시스템에 적용되어 다양하게 활용할 수 있을 것이다.

Deep Ensemble Methods for Food Ingredient Entity Recognition in Natural Language Processing

Kokoy Siti Komariah

Department of Artificial Intelligence Convergence, the Graduate School,
Pukyong National University

Abstract

In recent years, recipe-sharing websites are becoming popular among those who wish to learn how to cook or plan their menu. Individuals can choose ingredients that suit their lifestyle and health condition using online food recipes. The information from online recipes can be used to build various food, nutrition, and healthcare applications. However, the information collected from online food recipes lacks structured information. To extract such information into well-structured data, we can use a technique in natural language processing called Named Entity Recognition or NER. NER is a technique of recognizing key information or entities in a text and categorizing them into a predetermined category. However, three major issues arise when developing named-entity recognition in the food domain: (1) The availability of datasets for the food domain is still quite limited; (2) How to design a machine learning model that is effective and efficient in recognizing food entities; and (3) Existing NER models relied solely on a single model, and just

a few studies employ ensemble learning for NER, particularly none for the food domain.

This study aims to solve these problems via an ensemble learning technique, combining various learning algorithms to obtain a collective performance beyond existing models' performance. Drawing upon the ensemble technique, we propose a solution to the challenges mentioned above in two stages: first, we built an iterative self-training approach called SMPT (Semi-supervised Multi-model Prediction Technique). SMPT is a deep ensemble learning model that employs the concept of self-training and builds on 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 named the FINER dataset; and second, we proposed a food ingredient NER model called the Recurrent Network-based Ensemble model or RNE. RNE is a novel model for extracting food-related entities by incorporating deep ensemble learning with recurrent network models, including RNN, GRU, and LSTM. The experimental findings demonstrate that the proposed RNE model could extract information from food recipes more effectively than a single model. In future development, such information can support numerous food-related information systems.

Chapter I
Introduction

1.1 Background

The huge amounts of food data found on the internet provide a foundation for the development of artificial intelligence (AI) and contribute to the establishment of digital technology as an important part of food science and industries. Hence, to drive the development in the food domain, it is possible to replace every stage of this system, from food processing to food consumption, with a data-driven computational approach. However, these food data are still not being utilized to their full potential, and it is still difficult to meet the demand for efficient food data sharing, organization, and traceability which hinders the advancement of this domain. Organizing and integrating food data is important in food research and system development. Thus, developing a standard knowledge organization system for food, such as Food Knowledge Graph (FKG) proposed by Min et al. [1] in their research, is one step closer unveiled the full potential of food data utilization. Knowledge graphs offer a unified and standardized conceptual terminology presented in a structured form. As a result, they can properly organize the food data to benefit a wide range of applications effectively, as seen in Figure 1.1.

However, to construct a food knowledge graph, we need to extract the food information from those heterogeneous sources and find the important key information or entity relevant to our desired application. In natural language processing, or NLP, there is a subtask called named entity recognition or NER. NER intends to find a word or an expression that uniquely describes an element among a set of other elements with similar attributes. It provides a piece of rough categorical

information related to the target. Named entities in the text usually play vital roles in a sentence functionally and semantically. Named entity recognition is an information extraction technique that identifies keywords or information units dispersed within a text with known labels [2].

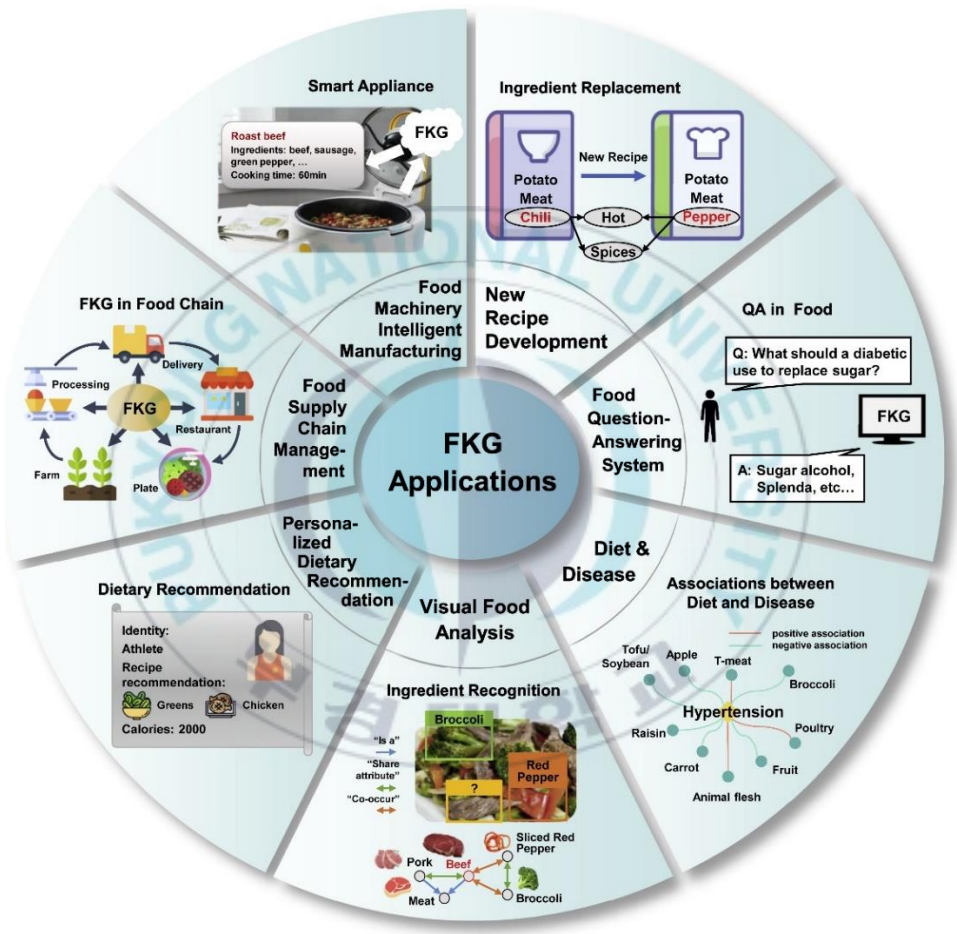


Figure 1. 1 Food Knowledge Graph (FKG) applications [1] contains various applications utilizing food data.

Thus, 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 build intelligent applications for the food industry, such as a personalized recommendation system for diets [3], [4], finding ingredient substitutes for people who are allergic to certain food or ingredients [5], or even calculating nutrient levels in food to prevent malnutrition [6]. However, when developing a reliable machine learning (ML) system, the biggest challenge comes from the need for extensive training data and a suitable model that can work perfectly with the data. Although numerous studies have been undertaken on creating food NER data [7]–[11] and related models [12]–[16], the food information domain is still relatively limited and therefore has much room for improvement.

1.2 Motivations

The importance of digital text data in food and nutrition has only recently drawn attention due to advance in food computing [17]. This development has brought a new dimension to food information processing. One technology that lies behind many applications and solutions is NER, a fundamental element for Natural Language Processing (NLP) in text. Three major issues arise when developing named-entity recognition in the food domain:

1) The availability of datasets for the food domain is still quite limited.

2) How to design powerful machine learning model that is accurate and efficient in recognizing food entities.

3) Existing NER models largely depended on a single model. There are only a few of studies utilizing ensemble learning for NER, and none for the food domain.

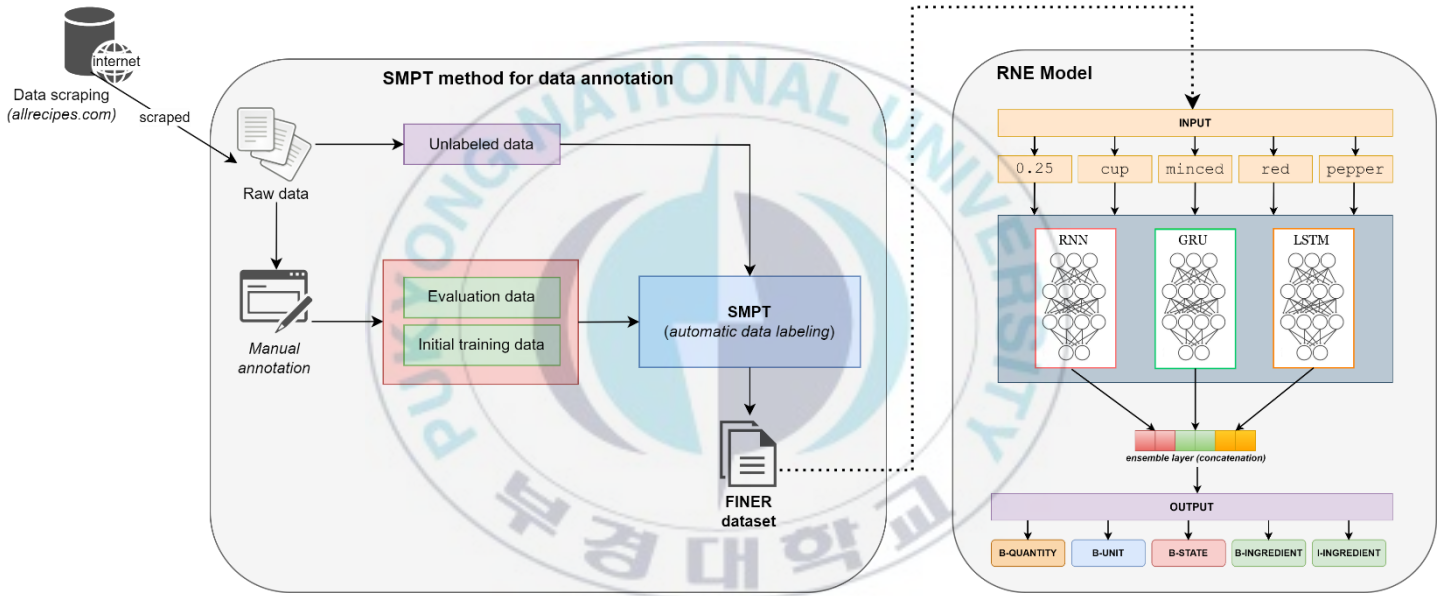


Figure 1. 2 Overall system framework

Based on these issues, we propose a strategy for developing NER datasets and model for the food domain that combines deep learning with ensemble learning. Figure 1.2 is the overall system framework of our study. First is the technique called Semi-supervised Multi-Model Prediction Technique or SMPT for constructing food ingredient named entity recognition dataset named FINER. SMPT utilized three pretrained language models with ensemble voting for the final decision on token's label. In the next study, this FINER dataset will be used as input in our second proposed method called Recurrent Network-based Ensemble model or RNE for short. RNE uses three recurrent network-based model in ensemble learning approach for predicting the entity's label in ingredient NER task.

1.3 Thesis Contributions

In order to take advantage offered by deep learning and ensemble learning. A deep ensemble model consisting of various deep learning algorithms was considered to extract food entities from text recipes. The contribution of the dissertation includes:

1) **The SMPT method**, a deep ensemble learning model that employs the concept of self-training that builds on pretrained language models (spaCy NER, BERT, and DistilBERT) in the iterative data labeling process with a voting scheme used as the final decision to determine the entity's label.
2) **The FINER dataset**, an annotated dataset for food ingredient entities. The dataset is made public and accessible on Figshare [18]: *https://doi.org/10.6084/m9.figshare.20222361.v3*.
3) **RNE model** [19], a novel model for extracting food-related entities by combining the deep ensemble method with recurrent network models such as RNN, GRU, and LSTM. According to the author's knowledge, this is the

first work to investigate a model that can benefit from an ensemble method based on deep learning used for food-related NER tasks.

4) In comparison to single models for food-related NER, the suggested NER task strategy yields superior results.

1.4 Outline of the Thesis

This section presents an outline of the contents of the thesis as shown in the relationship diagram in Figure 1.3, and each chapter can be explained as follows:

Chapter I Introduction: introduces the background of Named-Entity Recognition (NER), ensemble technique, and several deep-learning methods used in this thesis. In addition, this chapter also describes the research motivation, contribution, and the structure of this thesis.

Chapter II Literature Reviews: describes the basic knowledge including natural language processing, named-entity recognition, and the theory behind the proposed methods for extracting the entities such as self-training, transfer learning, transformers-based pretrained language model, and the RNNs-based model.

Chapter III Dataset Construction using SMPT method: presents the proposed method for constructing Food Ingredient NER dataset such as the data construction workflow, data preparation, NER labeling format, and our machine learning approach for annotating the dataset.

Chapter IV Enhancing Food Ingredient NER using RNE method: presents the proposed method to enhanced NER model's performance called Recurrent Network-based Ensemble (RNE) method including the deep learning classifiers used, the organization of the proposed model.

Chapter V Results and Analysis: validates the effectiveness of the proposed method for constructing the dataset using SMPT as well as evaluate the RNE model with various evaluation metrics such as Recall, Precision and F1-score.

Chapter VI Conclusion and Future Work: presents the conclusion of the thesis and some future works are described.

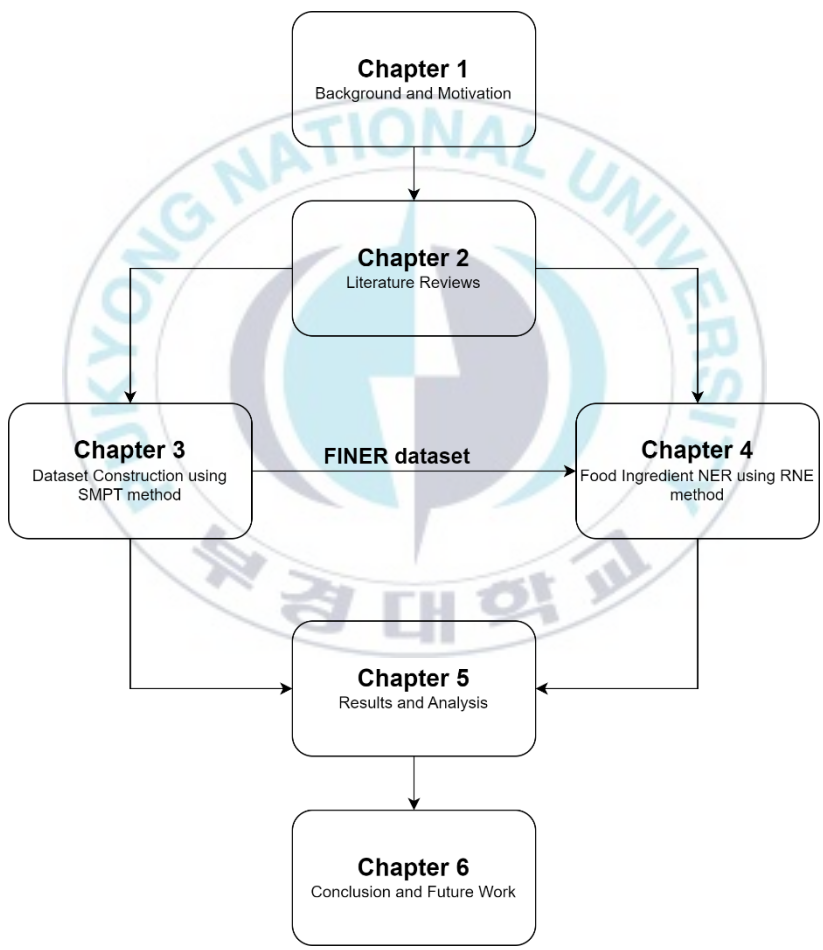


Figure 1.3 Relationship diagram between chapters.

Chapter II

Literature Reviews

2.1 Natural Language Processing

Natural Language Processing (NLP) is a subfield of Computer Science, Human language or linguistics, and Artificial Intelligence that focuses on the interaction between computers and human language as depicted in Figure 2.1 [20]. NLP is divided into two distinct subfields, namely Natural Language Understanding (also known as NLU) and Natural Language Generation or NLG, which respectively advance the goals of understanding and generating text [21].

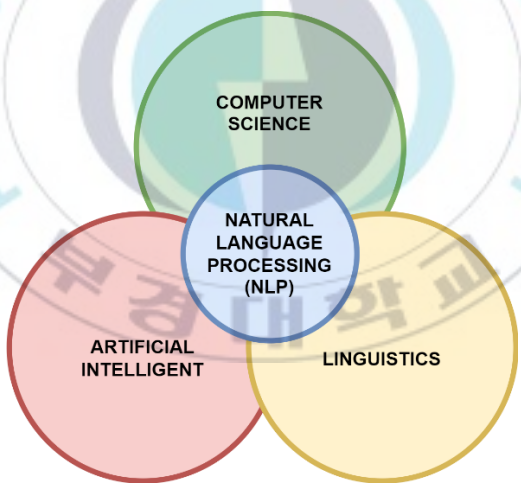


Figure 2.1 Natural Language Processing (NLP) is the intersection of computer science, artificial intelligence, and human language.

Figure 2.2 provides an overview of the various NLP categories. NLU and NLG are part of NLP. NLP aims to analyze and comprehend the text of a given

document, whereas NLU enables humans to have natural language conversations with computers. While both systems comprehend human language, NLU interacts with untrained humans to learn their intentions. In addition to recognizing words and interpreting their meaning, NLU is also designed to understand the meaning despite common human errors, including mispronunciations or transposed letters and words.

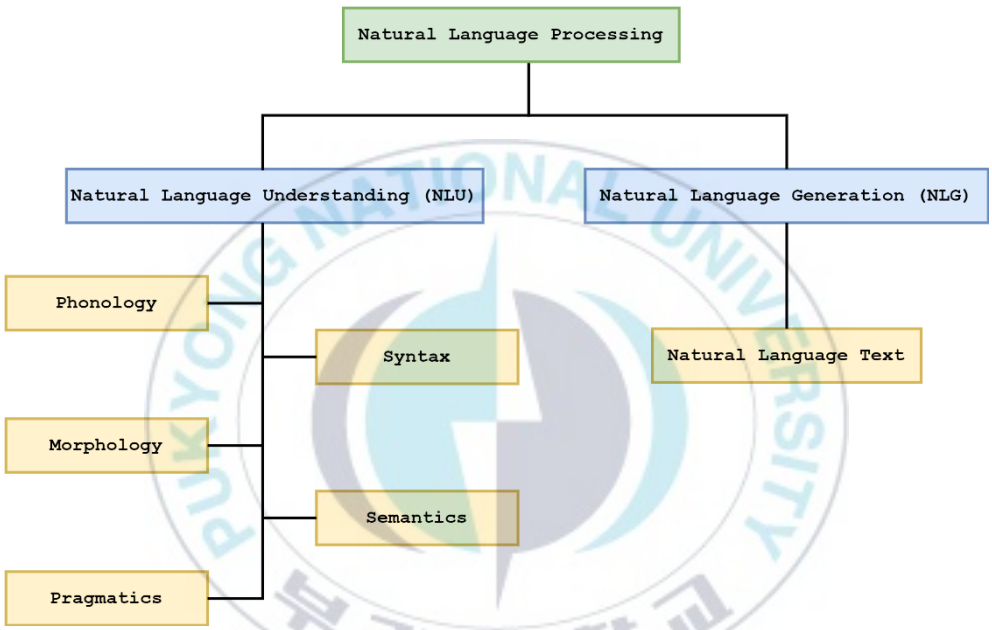


Figure 2. 2 General classification of natural language processing (NLP)

In contrast to traditional computer-generated writing, NLG enables computers to generate natural language text automatically, imitating how humans communicate naturally. In simple terms, computer-generated content lacks the fluency, emotion, and personality that make human-generated content engaging. NLG can utilize NLP so that computers can create human text in style replicating a human author. This human-like content is achieved by recognizing a document's main topic and applying

natural language processing to discover the optimal solution to compose the text in the user's native language.

In summary, NLP fills the gap between human language and computer comprehension. Typically, NLP algorithms evaluate large amounts of unstructured text data, such as documents, log files, and transcripts. Depending on the targeted outcomes, the output of an NLP model can differ. For instance, Amazon designed Alexa to recognize voice patterns, infer meaning, and do tasks to assist the users. As shown in Figure 2.3, the recent advancements in NLP and computational linguistics have allowed the NLP industry to expand from simple tools such as spell checking to more complex applications like as customer service chatbots, real-time voice-to-text translators, Google assistant, and others.

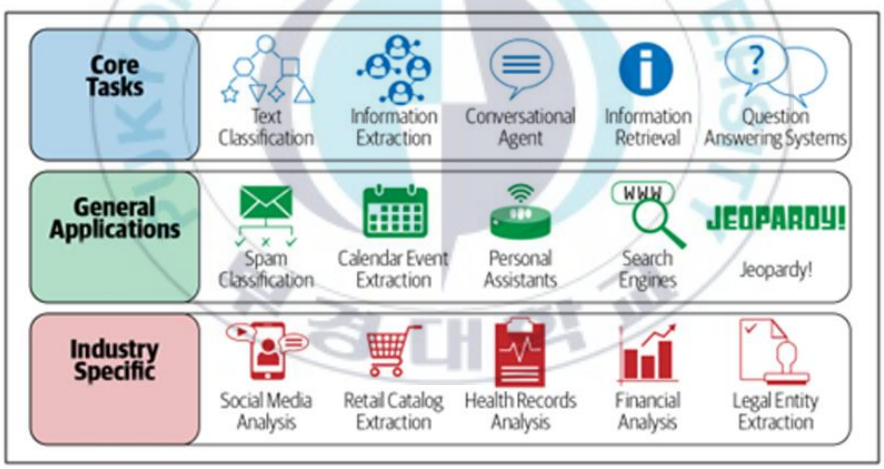


Figure 2.3 NLP tasks and applications [22].

2.2 Named Entity Recognition

Nowadays, people who are interested in learning how to cook or planning their menus have been referring more frequently to recipe-sharing websites for

inspiration. Likewise, in our daily meals, we expect to consume food and beverages with a complete nutritional content ranging from carbohydrates, proteins, vitamins, fats, and minerals. On the other hand, malnutrition is the root cause of a wide range of disorders, including anemia, sprue, goiter, as well as starvation and poor diet [23]. Therefore, it is necessary to be aware of food nutrition to have an accurate ingredient profile of food [6]. This ingredient profile will also provide information that will be helpful to individuals who follow a particular diet or have food allergies to certain ingredients or foods [3], [5]. One of the challenges those individuals face is finding food ingredients that perfectly fit the recipes. Due to various factors such as geography, climate, and time of year, certain recipe ingredients can be challenging to find or incredibly expensive in particular regions. As a result, we frequently search for alternative ingredients similar in taste, nutrition, and texture. Moreover, information regarding the ingredients of food recipes plays a critical role in the well-being of people who are following particular diets or who suffer from food allergies.

In recipes, a variety of cooking terminology are used. In addition to ingredients and utensils, cooking terms also include cooking actions and ingredient proportions. If we consider the cooking action of "*cutting*," there are a variety of cutting techniques, such as "*cutting into chunks*," "*slicing in rounds*," "*chopping into small pieces*," etc., depending on the cooking materials and purposes. If we can extract these cooking terms from recipe texts, we can use them to perform activities like extracting information from recipes and responding to questions. In this particular task, we make advantage of machine learning in order to perform automatic extraction of cooking terms. Among many different tasks that are involved in natural language processing, one of them is referred to as named entity recognition or NER for short.

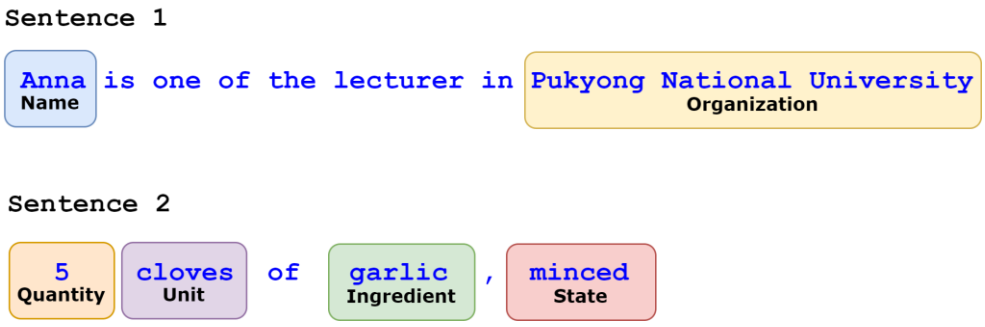


Figure 2.4 NER examples for general entities (sentence 1) and food ingredient entities (sentence 2).

Named entity is the method of extracting unique terms from natural language sentences, such as the names of people, places, and organizations, and many other [2]. This task can be expressed as a serial labeling problem. In sequential labeling for named entity recognition, input sentences are split into words, and each word is labeled with a name entity tag. The result of extracting labeled word strings is intrinsic expressions. In this study, instead of recognizing person and place names as intrinsic expressions, we train models to recognize the names of ingredients, product, quantities, unit, and cooking process or cooking state, as seen in the Figure 2.4. The figure presents the example of NER sentences with its recognized entities, sentence one contains general entities and sentence two contains food-related entities.

In addition, the existing NER systems do not necessarily rely on a single technique, sometimes it uses processing pipelines with a series of stages. Figure 2.5 presents a typical of NER process flow. The training set are transformed into a different features representation such as a vector space representation, on which base models are sometimes built with the assistance of external data. These models are applied to a testing set in order to evaluate their performance on previously unseen data [24]. According to Figure 2.5, the process starts with pre-processing by

preparing and refining the text to match the desired input. Next, employing NLP techniques for accurate sentence-splitting and POS tagging, and even adding annotation from other resources (e.g., dictionaries, gazetteers, Etc.) for better identification of entities. Later, several machine learning (ML) approaches are applied in parallel to label the entities. In the evaluation phase, we applied the model trained on the training set in the testing set to evaluate the model performance on unseen data.

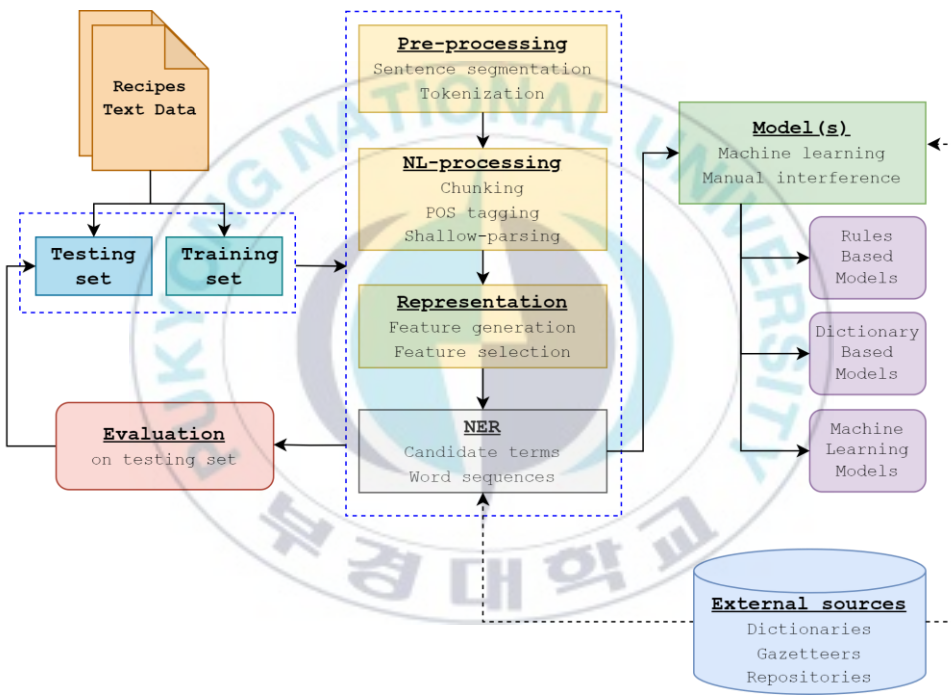


Figure 2.5 A typical NER process flow.

NER is useful in any scenario where a high-level overview of a significant amount of text is required. NER allows to easily understand the subject or theme of a text's content and group texts based on their relevance or similarity. Table 2.1 is an example of various NER application use cases.

Table 2.1 Various NER application use cases.

Use Case	Details
Human resources	▪ Accelerate the hiring process by summarizing applicants' resumes. ▪ Enhance internal procedures by classifying employee concerns and complaints.
Healthcare	▪ Improve the patient care standards and minimize workloads by obtaining essential information from laboratory reports. ▪ This is being done by Roche with pathology and radiology reports.
Content classification	Identifying the topics and themes of blog posts and news stories helps with content discovery and provides insight into trends.
Customer support/Help desk	▪ Improve response times by classifying user requests, complaints, and queries and filtering them using priority keywords.
Recommendation and search engine	▪ Enhance the rate and relevancy of search results and suggestions by summarizing descriptive language, reviews, and comments ▪ One prominent example of success in this field by using NER is Booking.com.
Education (Academia)	▪ Enable students and researchers to locate relevant information easier and faster by summarizing papers and documents and highlighting key terms, topics, and themes. ▪ For example: Europeana, the digital platform of the EU for cultural heritage, uses NER to make historical newspapers searchable.

2.3 Ensemble Deep Learning

The goal of an ensemble learning algorithm is to create a classifier with an improved prediction performance by aggregating the predictions of various learners as indicated in the Figure 2.6. Using ensemble methods, multiple learners are trained to solve the same problem. Unlike conventional learning approaches, which attempt to construct a single learner from training data, ensemble methods attempt to construct and combine multiple learners. There are a number of base learners within an ensemble. Base learners are typically generated from training data by the base classifier (base model), which could be logistic regression, random forest, neural networks, or other learning algorithms [25]. Various strategies to effectively combine base classifiers have been developed [26], [27]. The most common methods include bagging, boosting, stacking, voting, and blending [25], [28], [29].

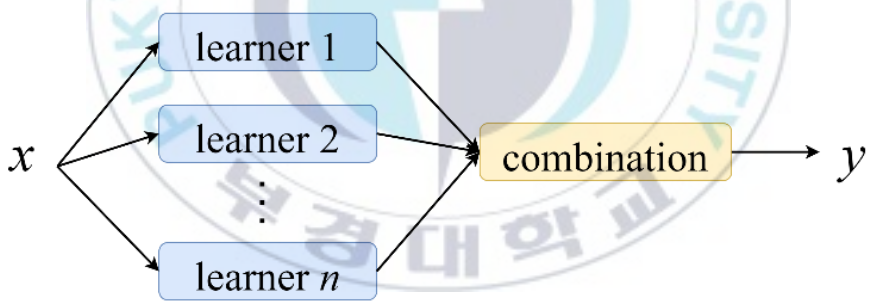


Figure 2.6 A typical ensemble architecture. x is input data (in our case is text), and y is the output prediction (in our case is class label or tag).

While deep neural network (DNN) architectures are now becoming increasingly popular and have demonstrated superior performance when compared to shallow or traditional models [30], [31]. DNN are a nonlinear method. They provide more flexibility and are able to scale proportionally to the amount of available training

data. The consequence of this flexibility is that they are sensitive to the specifics of the training data and could encounter different weights each time they are trained, leading to different predictions. In general, these are referred to as DNNs with high variance, making it challenging to create the final model for making predictions. In order to successfully reduce the variance of a DNN model, it is necessary to train multiple models and combine their predictions. This technique is known as ensemble learning, and it not only reduces prediction variance but also produces more accurate predictions than a single model. The combination of deep learning and ensemble method is called ensemble deep learning [32]. There are various ways to apply the ensemble method in deep learning. However, in general, the ensemble can be implemented in the following ways:

- **Varying the training data:** achieved by selecting different data to train each model in the ensemble.
- **Varying the models:** achieved by selecting a different subset of models to employ in the ensemble.
- **Combination:** achieved by selecting different combinations of ensemble member outputs.

As shown in Figure 2.7, they are various strategies for constructing ensemble deep learning from traditional techniques, including bagging, boosting, and stacking, into general and fusion techniques [32]. Various studies have been carried out by utilizing ensemble deep learning techniques, and it has been proven that the use of ensemble methods combined with deep learning can produce better model performance [32]–[37]. Therefore, out of these strategies, we utilized two of them for building our proposed methods which are majority voting and concatenation. In the following chapter, we will elaborate on each of our recommended methods in further detail.

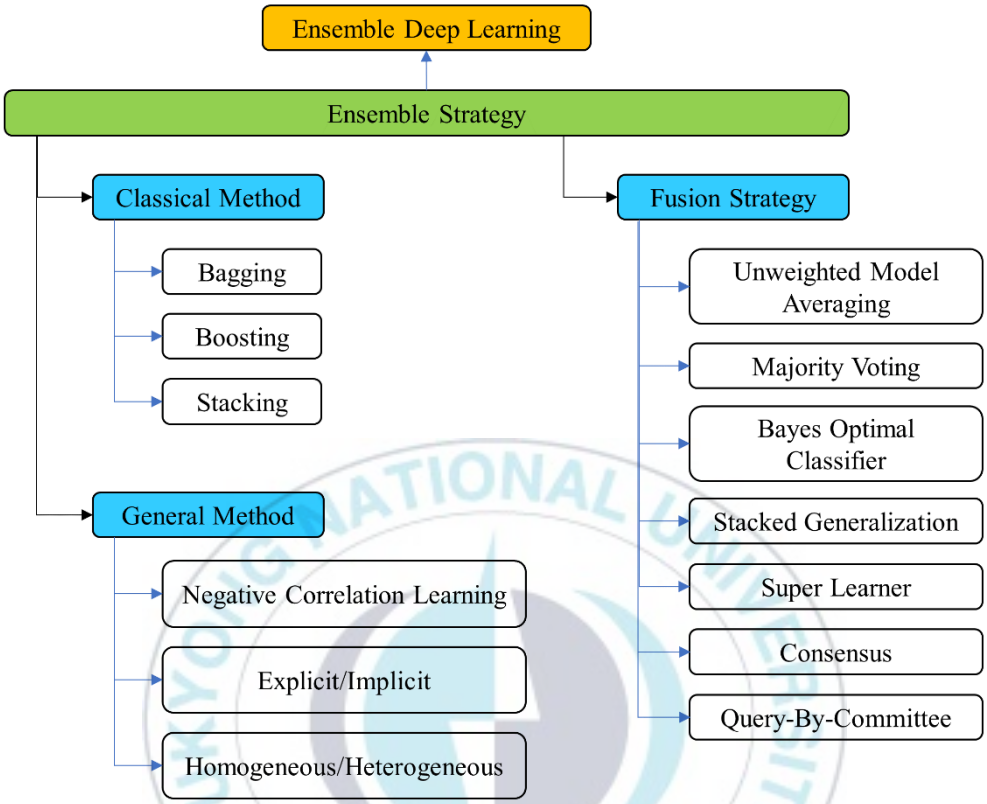


Figure 2. 7 Ensemble method strategies for deep learning [32].

2.4 Transfer Learning

In recent years, various sophisticated machine learning models for audio, image, and text data with large volume of data have been introduced. Unfortunately, the performance degrades whenever we train a model on one dataset and then attempt to apply it to a different dataset. This occurs because the model fails to generalize and comprehend the fundamental data patterns. Even a minor change in the data can throw it off guard. It is important to address this issue because real-world data are

constantly evolving, and it is impractical to constantly retrain a model from scratch for new scenarios. Transfer learning comes into play in this situation.

Transfer learning (TL) enables us to learn a task by employing labeled data from other tasks or domains that are related. The model's objective is to accomplish tasks. For example, recognizing the food entities in a recipe text, whereas the domain is the source of data. For example, all sample recipes are obtained from Allrecipes website. Figure 2.8 illustrates a comparison of TL with classic ML approach. Using TL approach, the knowledge acquired during task A for source domain A is saved and applied to the problem of interest in domain B [38].

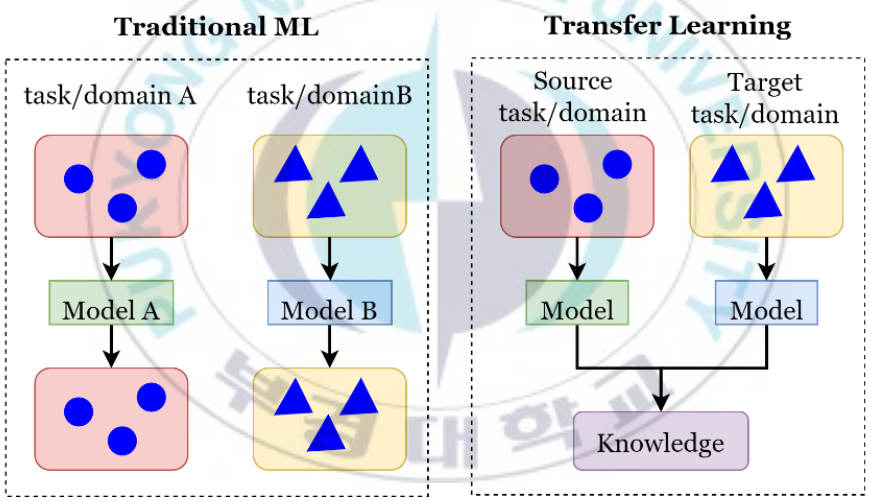


Figure 2.8 Comparison of Traditional ML and Transfer Learning.

Moreover, Figure 2.9 presents the concept of TL which involves utilizing the feature representation of a previously trained model so that a new model does not need to be trained from scratch. In ML, it is common to apply TL to NLP problems involving textual input or output. Typically, the pre-trained models are trained on large amounts of data that serve as a benchmark for the NLP frontier. The weights

derived from the models may be reapplied to other NLP tasks. These models can either be used directly to make predictions on new tasks or incorporated into the training process for a new model. Incorporating previously trained models into a new model reduces training time and generalization error. TL is particularly useful when the training dataset is small. In this instance, we initialize the weights of the new model with the weights of the pre-trained models.

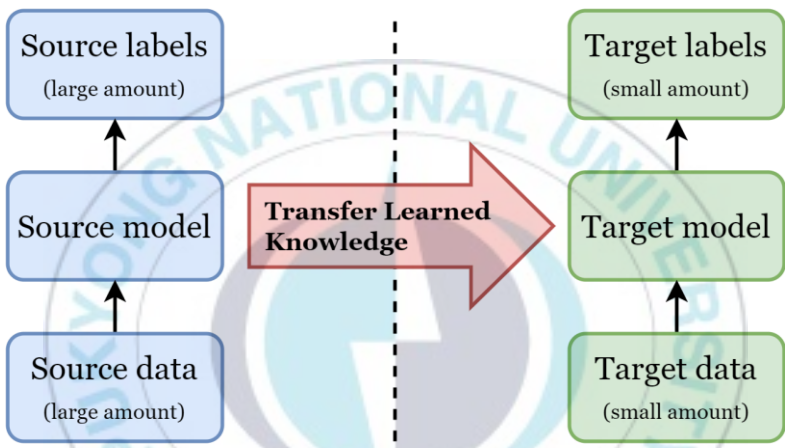


Figure 2.9 The concept of transfer learning (TL).

There are several types of TL that are common in modern NLP. They are distinguished across three dimensions based on: 1) whether the source and target settings deal with the same task; 2) the origin of the source and target domains; and 3) the order in which the tasks are learned [39]. Figure 2.10 explains a taxonomy that highlights the variations of TL in NLP.

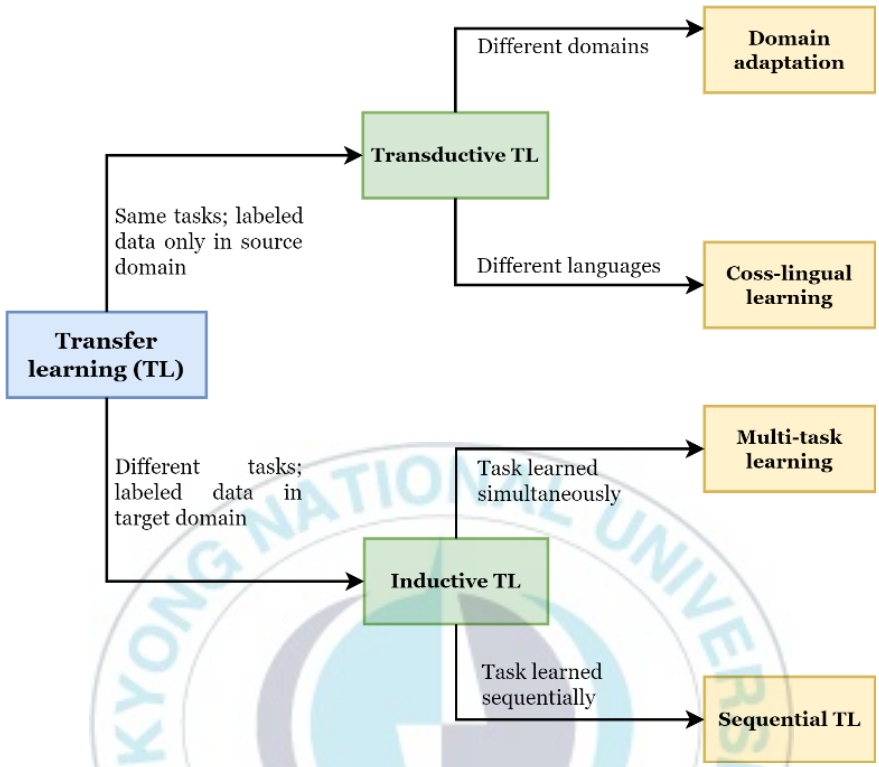


Figure 2.10 Transfer learning taxonomy in NLP.

Sequential transfer learning has produced the most significant gains to date. Figure 2.11 demonstrates the common practice of pretraining representations on a large unlabeled text corpus and then adapting these representations to a supervised target task using labeled data. In addition to Pre-training, another option in the TL step is Fine-tuning. In the standard transfer learning setup in Figure 2.11, a model is initially pre-trained on large amounts of unlabeled data using a language modeling loss such as MLM or masked language modeling [40]. However, in the fine-tuning procedure shown in Figure 2.12, using a standard cross-entropy loss, the pre-trained model is then fine-tuned using labeled data from a downstream task. While pretraining is computationally intensive, fine-tuning is relatively inexpensive [41]. Individual pre-trained models are downloaded, and fine-tuned multiple times based

on the target tasks/domains, which makes fine-tuning more important for the practical implementation.

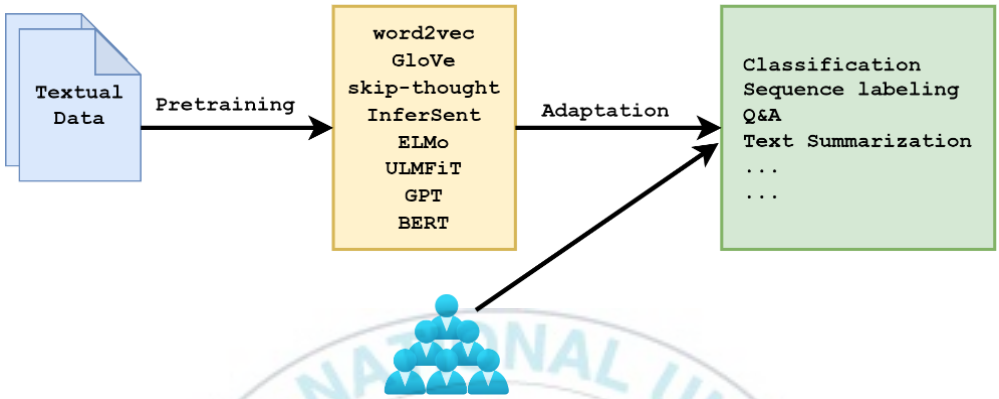


Figure 2.11 The standard TL procedure in NLP.

Figure 2.12 The procedure of Fine-tuning a pre-trained language model (LM).

2.5 Self Training

Self-Training is a type of Semi-Supervised ML where a model learns from both labeled and unlabeled data. Combining labeled and unlabeled examples, Semi-Supervised Learning increases the amount of training data available. As a result, we can improve model performance and save a substantial amount of time and money by eliminating the need to manually label thousands of examples [42].

Self-training employs a labeled training set L and an unlabeled data set U to train a model m. At each iteration, the model generates predictions $m(x)$ as a probability distribution over the C classes for all unlabeled examples x in U. If the probability assigned to the most probable class is greater than a predetermined threshold τ, x is added to the labeled examples with the pseudo label $p(x) = argmax\, m(x)$. This process is typically repeated for a predetermined number of iterations or until no more confident predictions can be made on unlabeled examples. This concept is illustrated in Algorithm 1 [43].

Algorithm 1 Self-training

repeat
  $m \leftarrow train\_model(L)$
  for $x \in U$ **do**
    if max $m(x) > \tau$ **then**
      $L \leftarrow L \cup \{(x, p(x))\}$
until no more predictions are confident

Figure 2.13 illustrates the steps of the Self-Training method. it starts with a set of labeled (L) for training the baseline model while sparing set of unlabeled (U) data for the next step. In the second step, we use the baseline model to predict the U. The part of results that meet our predefined criteria (e.g., prediction probability is greater than 90%) called pseudo-labeled data are combined with labeled data. And after that, we make new predictions and include newly selected observations in the pool of pseudo-labeled data. These steps are repeated until all data are labeled.

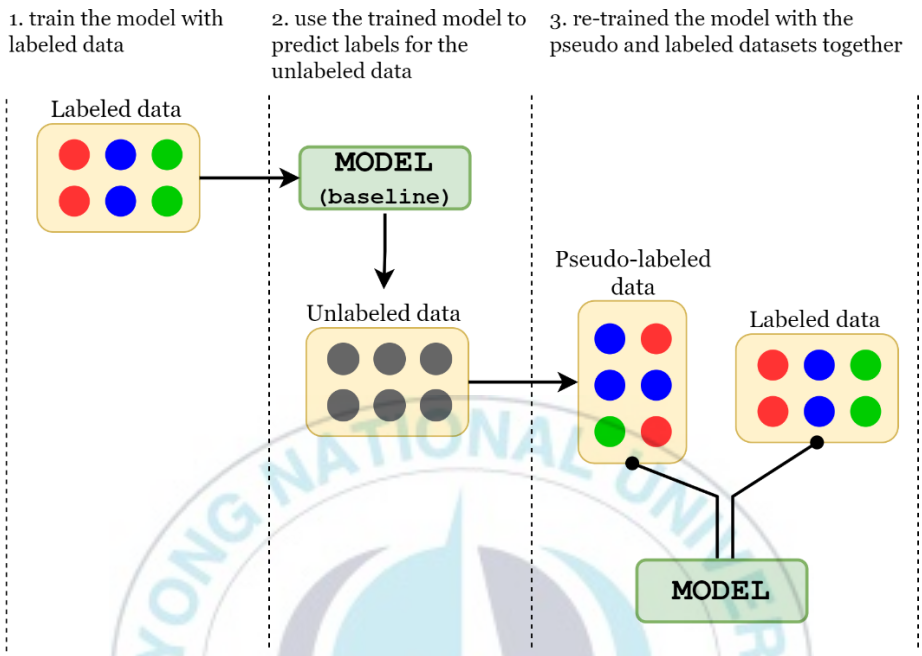


Figure 2.13 A step-by-step Self-Training procedure.

2.6 Deep Learning Approaches for NER

This section will presents a brief overview of the theory underlying the proposed methods for creating NER datasets and the algorithms used to develop the proposed deep ensemble model for food information extraction when addressing NER problems.

2.6.1 Transformer Model

The transformer model came from the research of Vaswani, et. al. in their paper titled "*Attention is all you need*" [44]. Transformer model utilized the encoder-decoder architecture that commonly used in Neural Machine Translation (NMT) models, but does not rely on recurrence and convolutions in order to generate an

output. Figure 2.14 shows the overall Transformer architecture, which employs stacked self-attention and point-wise, follow by fully connected layers for both the encoder and decoder.

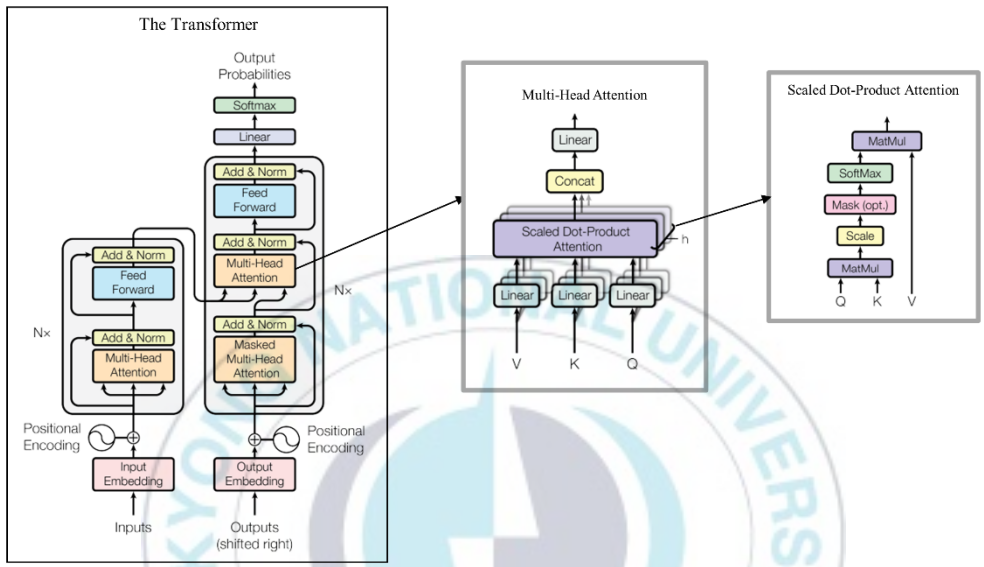


Figure 2.14 The transformer model architecture [44].

First part is Encoder, it provides an attention-based representation capable of locating a specific piece of information from a large context. The encoder consists of six identical layers where each layer has two sublayers including a Multi-Head Attention layer and a Fully Connected Feed-Forward layer (FFN). The first sublayer is the Multi-Head Attention layer. The Multi-Head Attention is a component of the Transformer that concatenates h distinct attention layers with different initializations [40], [44], [45]. The function of a Multi-Head Attention can be calculated as

$$Multihead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O \quad (1)$$

where, $head_1$ is i^{th} attention head which is given by:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (2)$$

and 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*
- $W_i^O \in \mathbb{R}^{hd_v \times d_{emb}}$

Here 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

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

The second sublayer is the fully connected feed-forward layer which consists of two linear transformations with ReLU activation in between [44].

$$FFN(x) = max(0, xW_1 + b_1) W_2 + b_2 \quad (4)$$

Each of the six layers of the transformer encoder applies the same linear transformations to all the words in the input sequence but uses different weight (W_1, W_2) and bias (b_1, b_2) parameters. In addition, unlike RNNs, transformer architecture cannot automatically determine where words are in a sequence. This information has to be added by incorporating positional encodings into the input embeddings.

The second part of transformer is decoder. Decoder has similar components with the encoder but with additional one sublayer called masked multi-head self-attention

layer. Unlike the multi-head self-attention layer, this sublayer is masked to prevent positions from attending to the future. While the encoder is designed to process each word in the input sequence regardless of its location, the decoder is adjusted to process only the prior words. Thus, the prediction for a word at position can only be based on the known outputs of the preceding words. This is obtained by applying a mask over the values produced by the scaled multiplication of matrices Q and K in the multi-head attention mechanism (which performs multiple single attention functions in parallel). This masking is conducted by eliminating matrix values corresponding to unauthorized connections.

Prior to the release of the transformer, the majority of cutting-edge NLP models were based on RNN. RNN processes data sequentially or word by word to reach the last word's cell. RNN is inefficient at handling long sequences, and the sequential structure of RNNs makes it difficult to maximize the performance of modern fast computing devices such as TPUs and GPUs. On the other hand, the transformers do not rely on past hidden states to determine word dependencies. They avoid recursion by processing sentences as a whole with the help of attention mechanisms and positional embeddings. Therefore, there is no possibility of losing or forgetting past information. Shortly after the introduction of the transformer model, many pre-trained language models that took advantage of the transformer model architecture became available and used to solve NLP problems, as an example is a decoder-only transformer which performs exceptionally well in language modeling tasks like GPT [46] and BERT [40].

2.6.2 Transformer-based NER

2.6.2.1 SpaCy NER

SpaCy is a Python and Cython-based open-source library for natural language processing that provides various NLP tools for such tokenization, POS-tagging, and Named Entity Recognition in text. Figure 2.15 shows the procedure that spaCy is used to train a custom NER model. It employs word embedding and a multilayer CNN with residual connections. It supports pretrained models in multiple languages and provides a default classifier for a wide variety of named or numerical entities, such as a person, organization, date, location, and event. In addition, it allows us to extend the NER model with new classes for novel entities.

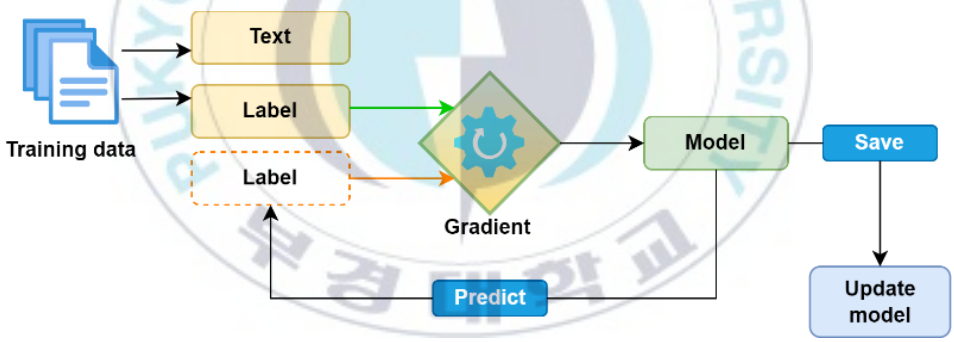


Figure 2.15 spaCy NER custom model training.

In this study, we design a neural network with a custom vector layer initialized with the pretrained spaCy's output layer. That layer is then trained using the spaCy library pre-training command [47], [48] on a domain-specific text corpus. Later, in the experiment, we also use the pretrained 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 pretrained model over the annotated data.

2.6.2.2 BERT

BERT is a language representation model introduced by Devlin et al. [40]. It uses stacked transformer encoders that learn a deep bidirectional representation from a large unlabeled corpus. An additional output layer is then added to fine-tune the representation in downstream NLP tasks. Fine-tuning involves modifying the neural network architecture slightly for improved predictions in target tasks while training the whole network. Pretrained BERT inherits the model weights learned during the pre-training, allowing downstream tasks to benefit from these powerful representations rather than learning from scratch.

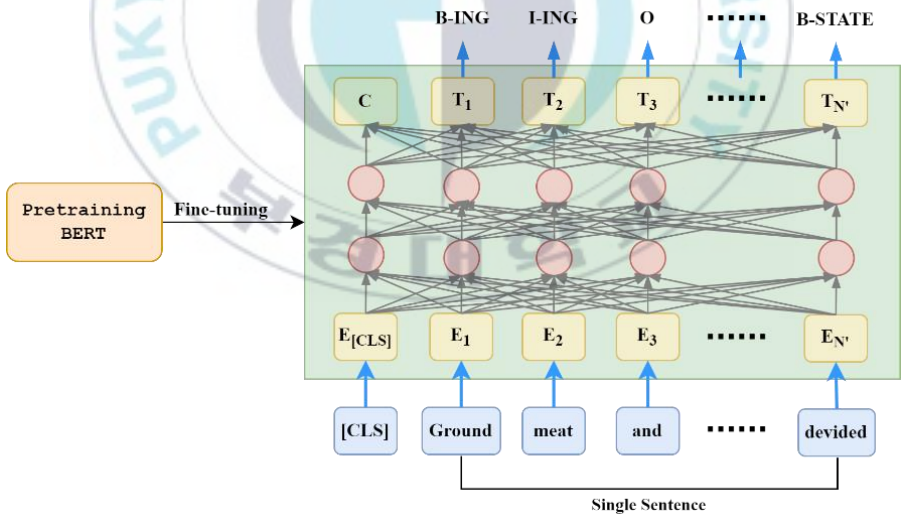


Figure 2.16 Fine-tuned BERT for FINER.

We use the weights of a pretrained BERT model (*bert-base-uncased*) to initialize the ingredient recognition task as shown in Figure 2.16. The BERT architecture is preserved, but the input and output are adjusted to our NER task.

2.6.2.3 DistilBERT

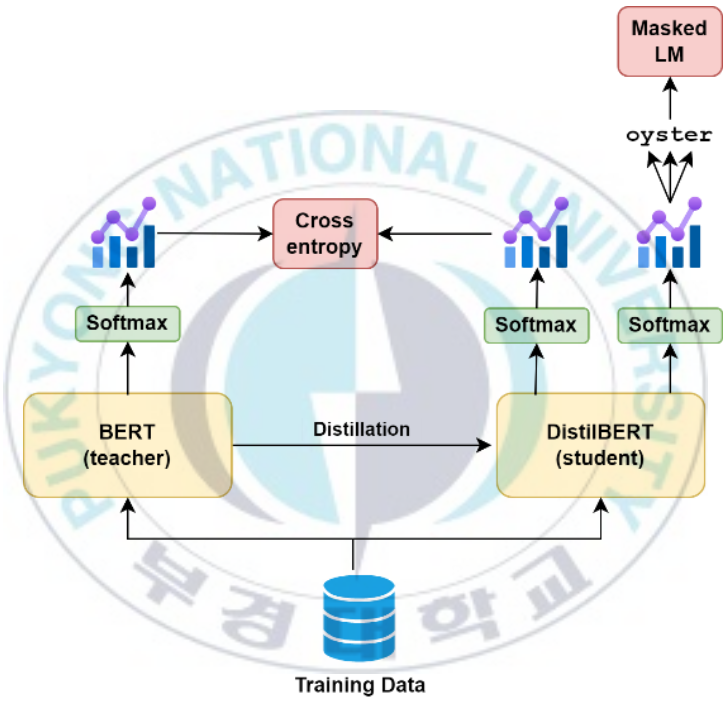


Figure 2.17 Knowledge distillation from BERT with the combination of cross entropy and the masked LM objectives.

DistilBERT [49] is a compact version of BERT and claimed to be lighter and faster than BERT with roughly comparable performance. It has 40% fewer parameters than *bert-base-uncased* and runs 60% faster at over 95% of BERT's performance as measured on the GLUE language understanding benchmark in

Victor Sanh et al., paper [49]. To reduce the computational requirements of modern large neural networks, DistilBERT uses a knowledge distillation technique known as teacher-student learning [50]. *Knowledge distillation* is a compression technique that involves training a small model to mimic the behavior of a larger model.

Figure 2.17, The masked language model (MLM) loss is used to train the student model, as well as 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 teacher's as possible. During teacher-student training, a student network is trained to replicate the distribution of a teacher network's total output or knowledge. This assists the student in generalizing in the same manner as the teacher. Instead of training with cross-entropy over hard targets (one-hot encoding of the gold class), we transfer knowledge from the teacher to the student using cross-entropy over soft targets (probability of the teacher). Thus, the loss becomes:

$$L = -\sum_i t_i * log(s_i), \tag{5}$$

where t represents the teacher's logits and s the student's logits. This loss results in a more robust training signal, as a single example imposes significantly more constraints than a single hard target. Hinton et al. introduce a softmax-temperature [51] to reveal more about the distribution's mass across the classes as follow:

$$\text{pi} = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)} \tag{6}$$

When T tends to 0, the distribution becomes a Kronecker (equivalent to the one-hot target vector), whereas it becomes a uniform distribution when $T \rightarrow +\infty$.

During training, the same temperature parameter is applied to both the student and the teacher, exposing more signals for each training example. In inference, T is set to 1, and the standard Softmax is recovered. Therefore, using teacher signals, it allows us to train a smaller language model, which was later called DistilBERT. In the implementation, we utilize the pretrained DistilBERT model (*distilbert-base-uncased*) in the same way we implement the BERT model.

2.6.3 Recurrent Neural Networks (RNNs) based NER

2.6.3.1 Recurrent Neural Network (RNN)

RNNs [19], [52] are type of artificial neural network whose memory may hold information about previous inputs in a sequence. The structure of basic RNN is present in Figure 2.18. In RNN, each word of the input sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ turns into vector form $\boldsymbol{y}_t$ by using the following equations:

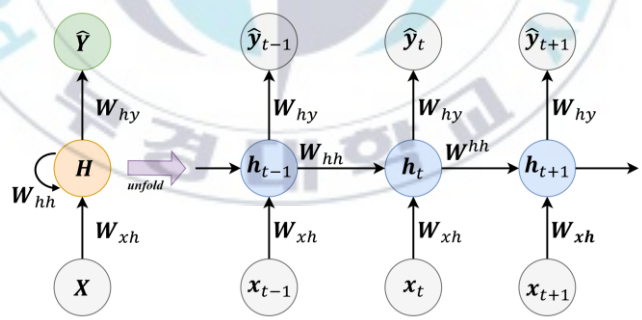


Figure 2.18 RNN architecture.

$$\boldsymbol{h}_t = \mathbf{H}(\boldsymbol{W}_{xh}\boldsymbol{x}_t + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h) \quad (7)$$

$$\boldsymbol{y}_t = \boldsymbol{W}_{hy}\boldsymbol{h}_t + \boldsymbol{b}_y \quad (8)$$

where $\boldsymbol{W}_{xh}$, $\boldsymbol{W}_{hh}$, and $\boldsymbol{W}_{hy}$ denote the weight matrices, h_t is the vector of hidden states that capture the information at time slice $t(t = 1,2,\ldots, T)$, $\boldsymbol{b}$ denote biases, and $\boldsymbol{H}$ is collection of activation functions for the hidden layer. A basic RNN with unidirectional flow transmits data from left to right with shared parameters are used for each time step. $\boldsymbol{W}_{xh}$, $\boldsymbol{W}_{hh}$, and $\boldsymbol{W}_{hy}$ are the same for each time step. When generating the prediction at time $\boldsymbol{t}$, it uses not only the current input $\boldsymbol{x}_t$ at time $\boldsymbol{t}$ but also the information from prior input at time $\boldsymbol{t-1}$ via activation parameter $\boldsymbol{H}$ and weights $\boldsymbol{W}$, which passes from the previous hidden layer to the current hidden layer. A drawback of RNN is that it can only make predictions based on preceding information. In this case, RNN utilizes information from earlier in the sequence to produce a prediction at a specific time, but not information given later in the sequence. To train RNN, Back-propagation through time (BPTT) is often used [53]. However, due to the gradient-vanishing and exploding problem, it is not practical to train standard RNNs with BPTT [54]. It is difficult to propagate errors from later time steps back to previous time steps enough to adjust network settings correctly. Thus, in the following development of the RNNs, the gated recurrent unit (GRU) and the long short-term memory (LSTM), have also been employed to resolve this concern [55]–[57].

2.6.3.2 Long-Short Term Memory (LSTM)

LSTM is an improved RNN type with memory cells [19], [57]. These memory cells are developed to handle long-term temporal dependencies in the data. It allows information to be added or removed from the current cell state. The input gate ($\boldsymbol{i}_t$), forget gate ($\boldsymbol{f}_t$), and output gate ($\boldsymbol{o}_t$) is computed to control this memory. Thus, LSTM units can propagate important features which appeared early in the input sequence over extended distances and capture potential long-distance relationships. LSTM, in contrast to RNN, contain three logistic sigmoid

gates and one tanh layer. Gates control the information that can passes through the cell. They evaluate which data is relevant for the next cell and which data can be disregarded. The output value normally falls between 0 to 1, where 0 indicates "*reject all*" and 1 indicates "*include all*." The architecture of the LSTM unit is visualized in Fig 2.19, and the formulas for calculating the time-step t for each hidden state in the LSTM unit are given in the following equations [58]:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i . [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i) \quad (9)$$

$$\boldsymbol{f}_t = \sigma\big(\boldsymbol{W}_f . [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f\big) \quad (10)$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o . [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o) \quad (11)$$

$$\widetilde{\boldsymbol{C}}_t = \tanh(\boldsymbol{W}_c . [\boldsymbol{h}_{t-1}, x_t] + \boldsymbol{b}_c) \quad (12)$$

$$\widetilde{\boldsymbol{C}}_t = \boldsymbol{f}_t * \boldsymbol{C}_{t-1} + \boldsymbol{i}_t * \widetilde{\boldsymbol{C}}_t \quad (13)$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t * \tanh(\boldsymbol{C}_t) \quad (14)$$

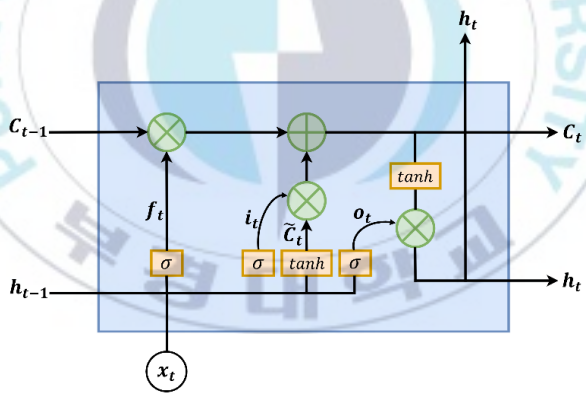


Figure 2.19 The LSTM unit architecture.

Each LSTM cell requires three inputs $\boldsymbol{h}_{t-1}$, $\boldsymbol{C}_{t-1}$, and input $\boldsymbol{x}_t$, as well as generates two outputs $\boldsymbol{h}_t$ and $\boldsymbol{C}_t$. $\boldsymbol{h}_t$ represent the hidden state, $\boldsymbol{C}_t$ represent the cell state or memory, and $\boldsymbol{x}_t$ represent the current data point or input for a given time $\boldsymbol{t}$. The first sigmoid layer ($\boldsymbol{\sigma}$) contains two inputs $\boldsymbol{h}_{t-1}$ and $\boldsymbol{x}_t$, where $\boldsymbol{h}_{t-1}$

is the previous cell's hidden state. It is known as the forget gate ($\boldsymbol{f}_t$) since its output determines how much information from the previous cell is included. The output is a number within the range [0,1] 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. $\boldsymbol{h}_{t-1}$ and $\boldsymbol{x}_t$. The tanh layer then creates candidate vector $\widetilde{\boldsymbol{C}}_{\boldsymbol{t}}$. Using a point-by-point multiplication of ($\boldsymbol{i}_t * \widetilde{\boldsymbol{C}}_t$), these two layers assess the data to be stored in the cell state. The result is then added to the previous cell state of the forget gate ($\boldsymbol{f}_t * \boldsymbol{C}_{t-1}$) to generate current cell state $\boldsymbol{C}_t$. Next, the cell's output is calculated using a sigmoid and tanh layer. The sigmoid layer determines the output cell state, whereas the tanh layer adjusts the output within the interval $[-1,1]$. Finally, a point-wise multiplication of these two layers produces the cell's output $\boldsymbol{h}_t$.

2.6.3.3 Gated Recurrent Unit (GRU)

GRU first designed by [19], [56] to allow each recurrent unit to capture dependencies at various time periods in an adaptive manner. It is similar to LSTM but has simpler cell architecture. GRU consists of gating units that control the information flow within the unit, but without memory cells. GRU computes the update and reset gates that control the flow of information across each hidden unit. Therefore, the update and reset gates, which are visualized by colored boxes in the GRU cell in Figure 2.20, can be calculated using the following equations [59], [60]:

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_{xz}\boldsymbol{x}_t + \boldsymbol{W}_{hz}\boldsymbol{h}_{t-1} + \boldsymbol{b}_z) \quad (15)$$

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_{xr}\boldsymbol{x}_t + \boldsymbol{W}_{hr}\boldsymbol{h}_{t-1} + \boldsymbol{b}_r) \quad (16)$$

$$\widetilde{\boldsymbol{h}}_t = tanh(\boldsymbol{W}\boldsymbol{x}\boldsymbol{h}\boldsymbol{x}_t + \boldsymbol{W}_{hh}(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}) + \boldsymbol{b}_h) \quad (17)$$

$$\widetilde{\boldsymbol{h}}_t = \boldsymbol{z}_t \odot \boldsymbol{h}_{t-1} + (1 - \boldsymbol{z}_t) \odot \widetilde{\boldsymbol{h}}_t \quad (18)$$

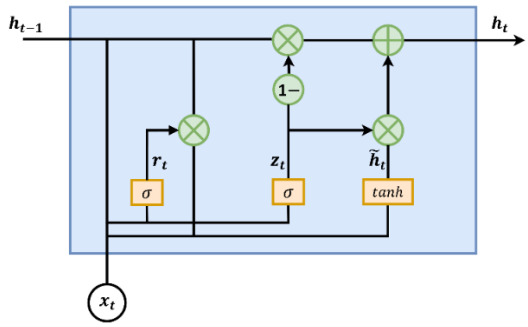


Figure 2.20 The GRU unit architecture.

where $\boldsymbol{z_t}$ represent update gate, $\boldsymbol{r_t}$ represent reset gate, $\boldsymbol{W}$ represent the weight vector, $\odot$ is an element-wise multiplication and σ is the sigmoid function. At each time $\boldsymbol{t}$, it takes input $\boldsymbol{x_t}$ and hidden state $\boldsymbol{h_{t-1}}$ from the previous time $\boldsymbol{t-1}$. Then, the outputs are a new hidden state $\boldsymbol{h_t}$ for the next time t. In order to locate $\boldsymbol{h_t}$, two steps are needed in GRU. First, the candidate hidden state is generated. It multiplies the input and hidden state from the previous time $\boldsymbol{t-1}$ by the reset gate output $\boldsymbol{r_t}$. This information is sent through the tanh function, which returns the hidden state of the candidate. The reset gate value in this equation determines how much the previous hidden state influences the candidate state. If $\boldsymbol{r_t}$ = 1, all information from $\boldsymbol{h_{t-1}}$ is considered. Otherwise, if $\boldsymbol{r_t} = 0$, the information from the previous hidden state is ignored. Once the candidate state is obtained, it is used to generate $\boldsymbol{h_t}$, which involves the update gate. In GRU, instead of utilizing a separate update gate like in LSTM, both the historical information ($\boldsymbol{h_{t-1}}$) and the new information from the candidate state are controlled by a single update gate. Moreover, $\boldsymbol{z_t}$ is crucial in this equation. It can determine how much information from the past must be passed to the future. The $\boldsymbol{z_t}$ value within range of 0 to 1.

Chapter III
Food Ingredient Named-Entity Data Construction using Semi-Supervised Multi-Model Prediction Technique

3.1 Background and Related Works

There have been several studies on food data construction being conducted with various approaches and data sources [17]. Table 3.1 compares the existing food datasets and the FINER dataset, which has been developed by the proposed method. Among the datasets in Table 3.1, Recipe1M+ [10] and RecipeNLG [9] are containing both image and text, they both used for multimodal machine learning tasks. However, the goal of our study is extracting named entities from food recipe text. There are datasets available for task, but they are usually small in terms of the availability of training samples [12].

Table 3.1 Available food dataset compared to our FINER dataset.

Dataset	Method	Source	Dataset Size (recipes)	Entities
FoodBase [7]	Ruled-based approach	Allrecipes	1000 curated and 21.790 uncurated version	Based on Hansard corpus semantic tags: AG (food and drink) AE (animal) AF (plant)
Recipe1M+ [10]	Deep learning approach	Various cooking sites and image search engines	1 million recipes and 13	-

Dataset	Method	Source	Dataset Size (recipes)	Entities
		for image data extension	million food images	
RecipeDB [8]	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 [9]	Deep learning approach	Recipe1M+ and auhtors private data gathered from various cooking sites	Over 1 million new data	-
TASTEset [11]	Deep learning approach	AllRecipes Food.com Tasty Yummly	700	Food Quantity Unit Process Physical Quality Color Taste Purpose Part
FINER (Our)	Deep learning approach	Allrecipes	64.782	Ingredient Product Quantity Unit State

The problem of limited data on food and its attributes has recently been addressed in several papers with proposals for building food NER datasets [7], [8]. Batra et al.

presented [8] while Popovski et al. discussed the FoodBase corpus, which was developed using a rule-based approach [7]. While easily available, it is rather small, consisting of 1000 recipes for the curated version (manually evaluated) and 21.790 for the uncurated version. Similar to FoodBase, a recently published dataset called TASTEset [11] is an even smaller set of 700 recipes. On the other hand, the RecipeDB dataset, built by Batra et al.,[8] is relatively large and covers a wide range of recipes. It has been derived from numerous online food recipe sharing sites. Unfortunately, the dataset cannot be use directly because it is intended for search applications about recipes or food ingredients.

This study focuses on extracting various entities in the recipe texts from the Allrecipes website, a popular online social media for sharing food recipes. In particular, we propose a novel iterative framework to build a new dataset of annotated ingredient entities in various recipes called Food Ingredients Named Entity Recognition (FINER). This study proposes a generic method for building a food NER dataset using ML techniques to address the data limitation issue called Semi-supervised Multi-model Prediction Technique (SMPT). SMPT is an ensemble deep learning model which employs the concept of self-training that builds on pretrained language models (such as spaCy NER [47], BERT [40], and DistilBERT [49]) in the iterative data labeling process [61] with a voting scheme used as the final decision to determine the entity's label.

3.2 Data Construction Workflow

The data construction workflow in our study consists of four stages: (a) Data preparation; (b) Manual data annotation; (c) Model training and automatic labeling; and (d) Dataset evaluation. Several NLP libraries, including SpaCy [47], NLTK [62], and Doccano annotation tools [63] were utilized throughout the building process.

The detailed data construction process shown in Figure 3.1 involves the following four steps:

a) First stage is data preparation. We begin by cleaning the text data collected from the Allrecipes website, followed by a number of preprocessing steps.
b) The second stage is the manual data annotation. We begin with a small set of 2,000 manually labeled data for initial training set and evaluation set (each of 1000 data). The first training set is utilized to create a baseline NER annotator, while the evaluation set is held for the final evaluation of the complete dataset.
c) The third stage is model training and automatic labeling. In this stage, baseline models are built utilizing the initial training set from the previous step. To annotate and generate our dataset, we propose a semi-supervised technique named Semi-Supervised Multi-Model Prediction Technique (SMPT). We will explain in detail about the SMPT method in Subchapter 3.5. After baseline model is created, we then used this model to predict labels for the remaining unlabeled set. As of now, 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.
d) The final stage is dataset evaluation. After multiple iterations, the Food Ingredient NER (FINER) dataset is obtained. Using a set of classifiers including CRF, Bi-LSTM, and BERT, we evaluate the dataset's quality indirectly. We assess their performance using the reserved evaluation set.

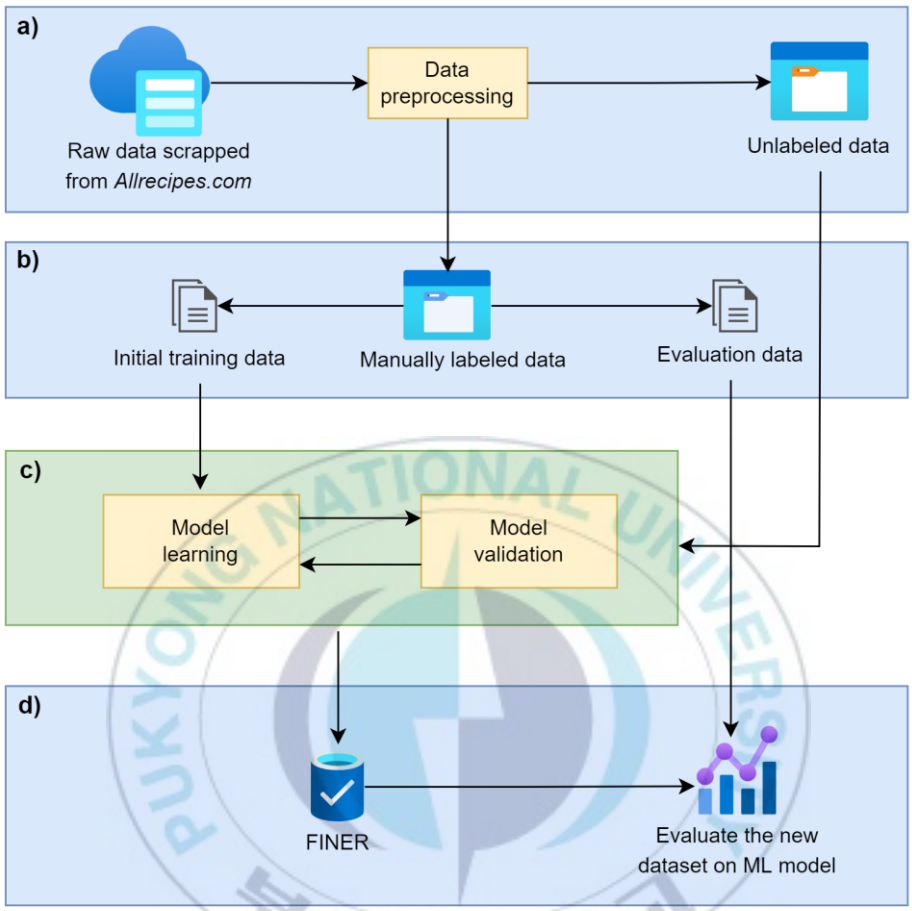
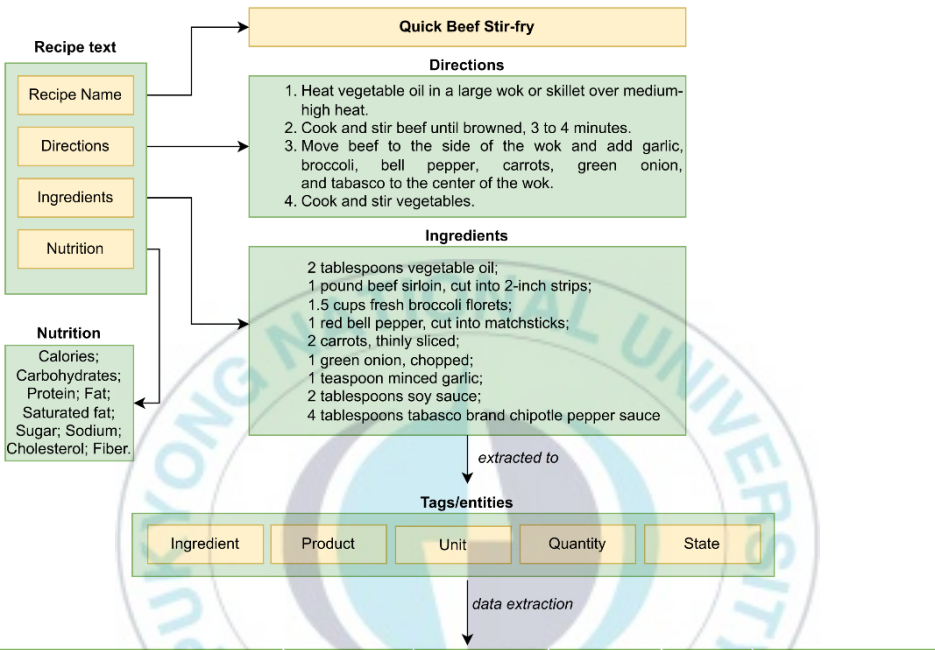


Figure 3.1 Dataset construction flowchart.

3.3 Data Preparation

The dataset in this study is based on the recipe text scraped from the *Allrecipes* website [64]. The structure of the data is illustrated in Figure 3.2 which consists of recipe names, ingredients, direction, and nutrition. However, this study extracts ingredient entities and their attributes, including ingredient name, product, quantity, unit, and ingredient state, from the ingredient section. Figure 3.3 shows the recipe

data stored in csv file containing columns with information about recipes. After data preprocessing, we parse and convert the data to the NER CoNLL format shown in Table 3.2 and save it as a CoNLL file shown in Figure 3.4 as an input to the model.



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 3.2 A recipe structure and ingredient data extraction workflow.

name	category	rating	ingredients	directions	calories	carbohydrat	sugars	fat_g	cholesterol	protein	dietary_fiber	sodium_mg
Garlic and Herb Marinade	side-dish	4.55	1/4 cup water ; 1/8 cup vinegar	In a medium bowl	85.5	1	0	9.2	0	0.2	0.3	291.5
Wisconsin Bratwurst	meat-and-poultry	4.53	2 pounds fresh bratwurst sausa	Prick bratwurst wi	557.8	10.4	0	44.9	116	13.7	0.1	907.6
Turkey Brine	side-dish	4.83	1 gallon vegetable broth ; 1 cup	In a large stock pot	2.8	0.6	0	0.1	0	0.1	0.4	5640.3
Tainted Fruit Shots	drinks	4.9	1 (6 ounce) package fruit flavor	In a large bowl, mi	66	7.6	0	0	0	0.7	0	22.5
Oyster Shooter	seafood	3.86	4 shucked oysters ; 4 teaspoon	Place each oyster i	34.6	0	0	0	0	0.1	0	29.9
Argentinean-Style Ribs	world-cuisine	4.1	1 cup coarse salt, or as needed	Preheat an outdoc	1760	0	0	164.3	344.7	65.3	0	15404
Beer Cheese Dip I	appetizers-and-snacks	4.2	1 (8 ounce) package cream che	In a medium bowl	67.3	1.3	0	5.6	16.3	2.6	0	119.3
Beer Dip II	appetizers-and-snacks	3.63	2 (8 ounce) packages cream che	In a medium bowl	74.4	1.6	0	6.5	20.5	1.5	0	137.5

Figure 3.3 An example of data scrapped from *allrecipes.com*.

Table 3.2 An example of a sentence in an NER CoNLL format.

Sentence	Word	Tag
Sentence #1	1	B-QUANTITY
Sentence #1	slice	B-UNIT
Sentence #1	whole	B-INGREDIENT
Sentence #1	wheat	I-INGREDIENT
Sentence #1	bread	I-INGREDIENT

```
1           B-QUANTITY
slice       B-UNIT
whole       B-INGREDIENT
wheat       I-INGREDIENT
bread       I-INGREDIENT

0.5         B-QUANTITY
(           O
1           B-QUANTITY
ounce       B-UNIT
)           O
package     B-UNIT
dry         B-INGREDIENT
ranch       I-INGREDIENT
-           I-INGREDIENT
style       I-INGREDIENT
dressing    I-INGREDIENT
mix         I-INGREDIENT
```

Figure 3.4 An example of an NER CONLL file format.

Overall, after data preprocessing, the dataset contains 64,782 raw recipes, a total of 1,397,960 words, and 181,970 sentences (ingredient phrases), as detailed in Table

3.3. In addition, the recipe text dataset has eight food categories such as breakfast and brunch, dinner, main dishes, side dishes, drinks, dessert, bread, and salad. It has been cleaned up so that the data's format and structure match to the desired input. The preprocessing comprised of several steps, starts by removing all special characters, white spaces, stop words, punctuation symbols, tokenizing the data, converting it into lower-case, followed by lemmatization. In addition, the following rules are defined:

- Since our dataset is made up of a list of ingredients, stop words and punctuation may not always be meaningless to the text's intent, but it may help interpret entities. Therefore, we have constructed custom lists of stop words and punctuation. For example, the sentence *"1 (2 ounces) package butter"* means 1 package of butter equals 2 ounces. Thus, the parentheses are kept because they contain information for converting the amount of ingredient to standard units.
- Standardized the unit and quantity measurements. 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 *"½"* becomes *"0.5"*.
- To simplify the extraction, every phrase in the ingredients section is split into individual sentences. In our experiment, only the ingredients section was employed. This section contains all the ingredients for a particular recipe. Figure 3.5 demonstrates the splitting of each list into ingredient phrases. After the pre-processing procedure, the final dataset contains 181,970 ingredient phrases or sentences.

Table 3.3 The dataset information details.

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

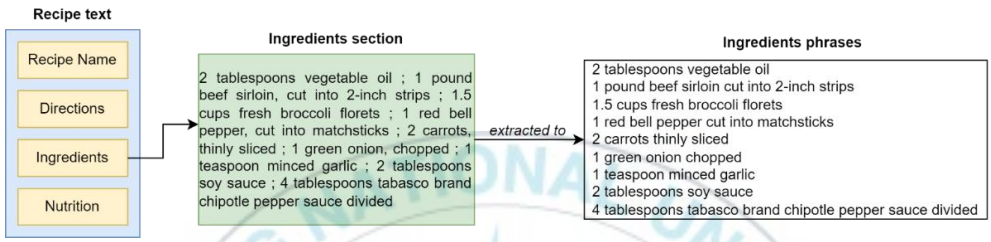


Figure 3.5 Splitting process of ingredient lists into individual phrase.

After a series of pre-processing steps, the recipes text data is divided into three sets as summarized in Table 3.4. For training and evaluation data, we manually annotated the first two sets using Doccano annotation tools [63] and left the remaining set unlabeled, which will be labeled recursively by the proposed method of NER annotator.

Table 3.4 The organization of the initial dataset. Initial training set and evaluation set are manually annotated.

Initial training set	1,000
Evaluation set	1,000
Unlabeled dataset	179,970
Total	181,970

3.4 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. This research aims to extract food ingredient entities and its attributes from recipe texts. This study defines five different entity classes, each of which corresponds to an entity tag listed in Table 3.5. To chunk the entity word, we employed the IOB2 format [2], [65]. 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 3.6 presents a details explanation of the IOB format. In this format, a tag is prefixed by one of B, I, and 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.

Table 3.5 Entity classes with their respective definitions and examples.

Class	Description	Example
INGREDIENT	Name of the food or ingredient.	Carrots, garlic, vegetable oil, etc.
QUANTITY	Measurement unit.	Gram, pound, tablespoon, etc.
UNIT	Measurements of the food or ingredient associated with the unit.	1 ½, 25, 0.5, etc.
PRODUCT	Food or ingredient from specific brand mention.	Tabasco brand chipotle pepper sauce, Archer farms dark chocolate hot cocoa mix, etc.

Class	Description	Example
STATE	Processing state of the food or ingredient.	Minced, chopped, cut into 2-inch strips, etc.

Table 3.6 IOB tagging scheme.

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

3.5 Semi-Supervised Multi-Model Prediction Technique (SMPT)

Figure 3.6 depicts a general procedure of our proposed method for ingredient named entity data annotation namely Semi-Supervised Multi-Model Prediction Technique or SMPT. SMPT is an iterative step similar to the bootstrapping method used by Kim et al. in [66]. However, instead of using CRF for bootstrapping and automatic labeling the unlabeled data. We adopted the concept of self-training in the data labeling process. Given the small labeled datasets, we train a baseline classifier based on pretrained 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 set of labeled data is incorporated into the input for the next iteration. The whole procedure is repeated until the unlabeled sample is labeled.

A key component in the proposed method is the development of the classifiers that will be trained in this section. The SMPT training process consists of two main

components: the training process itself and the dataset labeling scheme with the dataset growth factor. Thus, we will explain these two components in the following sections.

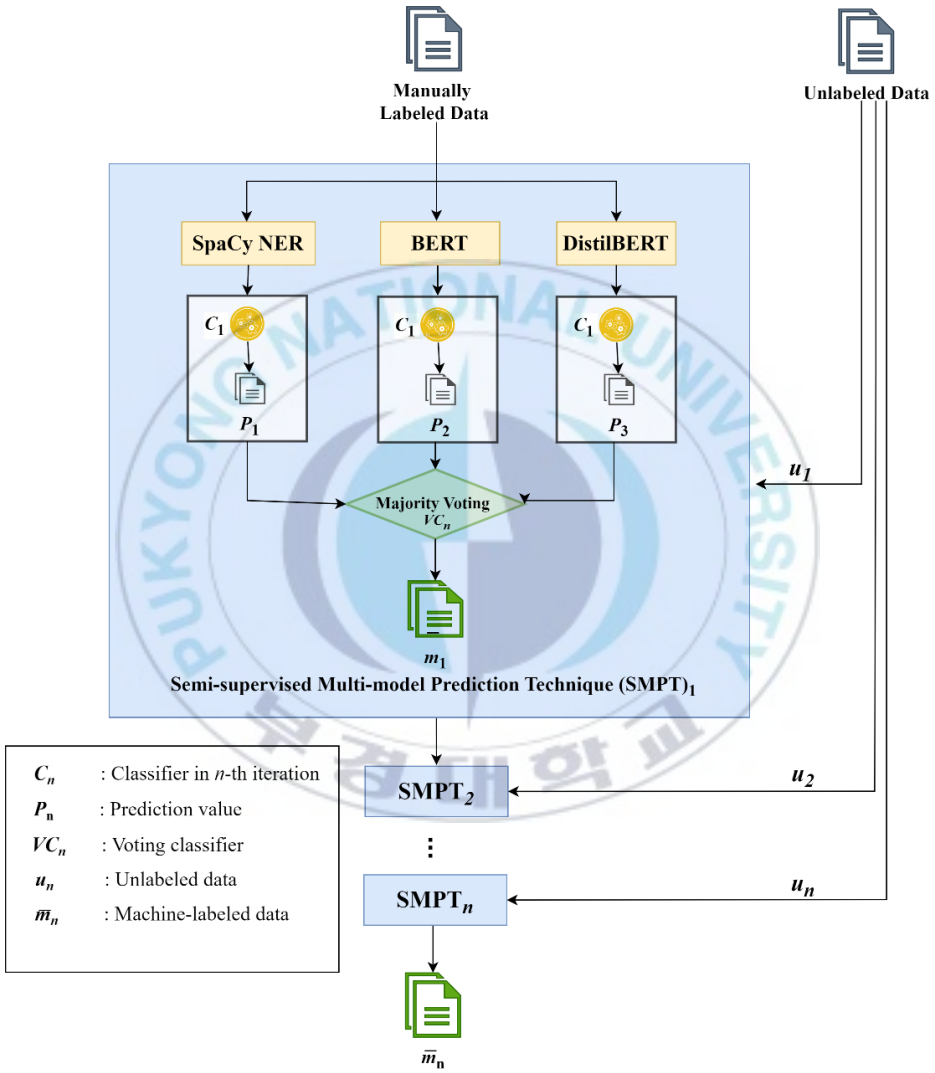


Figure 3.6 The SMPT method workflow.

3.5.1 Training

The core tasks of classifier training and data labeling in SMPT consists 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. The final decisions on the unlabeled tokens are made by the majority voting scheme:

$$\bar{m} = argmax_{m \le M} \sum_{c=1}^{C} d_{c,m} \quad (19)$$

where $\bar{m}$ is the final prediction label (class), M is number of classes, $C = |C|$ 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 no tokens are left unlabeled.

3.5.2 Dataset Building Schemes

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 decided 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 3.7 explain the data size growth procedure for data training and labeling using scheme 1 with growth factor 2 as an example. According to Table 3.7, we have an initial training set that we have prepared in Section 3.3 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 $u_1 = s * t_1$ for a promotion to the training set of size $t_2 = t_1 + s * t_1$ for the next round. Therefore, the size of the unlabeled data 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 the entire data are assigned and labeled. An example of this procedure for the growth factor $s = 2$ is demonstrated in Figure 3.7.

Table 3.7 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 scheme 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	$t_4 = t_3 + 2t_3$	$u_4 = 2 * t_4$
.	.	.
.	.	.

Iteration	The set	Scheme 1 ($s = 2$)
n	$t_n = t_{n-1} + 2t_{n-1}$	$u_n = s * t_n$

Initial data = 1000
Total Unlabeled data = 179,970

Iteration	Scheme 1 ($s = 2$)	
	Training set size	Unlabeled set size
1	$t_1 = 1{,}000$	$u_1 = 2 * 1{,}000 = 2{,}000$
2	$t_2 = 1{,}000 + 2{,}000 = 3{,}000$	$u_2 = 2 * 3{,}000 = 6{,}000$
3	$t_3 = 3{,}000 + 6{,}000 = 9{,}000$	$u_3 = 2 * 9{,}000 = 18{,}000$
4	$t_4 = 9{,}000 + 18{,}000 = 27{,}000$	$u_4 = 2 * 27{,}000 = 54{,}000$
5	$t_5 = 27{,}000 + 54{,}000 = 107{,}000$	$u_4 = 99{,}970$ (*remaining data*)

Figure 3.7 An example of data size growth procedure with growth factor $s = 2$.

Chapter IV
Ensemble-based Recurrent Networks for Food Ingredient Named Entity Recognition

4.1 Background and Related Works

This section will provide a brief overview of the background on developing NER for food information extraction, including various studies employing the recurrent networks model and ensemble methods to address NER challenges.

4.1.1 Food-Related NER

NER tools and frameworks use various approaches that can be classified into three categories: machine learning, rule-based, and dictionary-based [67]. In the food domain, there have been a few studies on the NER problem. In the ruled-based approach, there are FoodIE [14] and drNER [68]. In addition, several databases for food-related NER have also been developed, such as FoodBase [7] and RecipeDB [13]. FoodBase raw data were obtained from the Allrecipes website, while RecipeDB was from Allrecipes and Food.com websites. After the establishment of the FoodBase corpus, the most recent study in [12] evaluated four distinct NER techniques using this dataset, such as FoodIE, NCBO (SNOMED CT), NCBO (OntoFood), and NCBO (FoodON). Moreover, to enhance the performance of the FoodBase corpus, FoodNER [16] and BuTTER [15] were proposed. FoodNER uses a bidirectional encoder representation from the transformers (BERTs) model to extract food entities, whereas BuTTER uses bidirectional long short-term memory (BiLSTM) and conditional random fields to extract food entities (CRFs).

4.1.2 Ensemble Method for NER

Ensemble learning combines several base classifiers to obtain better generalization performance. Diverse methods for effectively combining base classifiers have been proposed [26], [27]. Voting, bagging, boosting, blending, and stacking are the most frequent approaches [25], [28], [29]. In the NER task, several research studies have used the ensemble method. These studies are summarized in Table 4.1. Voting has been a preferred method for most NER investigations among the ensemble strategy [69]–[75]. *Voting* is an ensemble method that combines the performance of multiple classifiers and selects a class with the most votes [27]. Several of the studies in Table 4.1 employ several strategies, including concatenation [76], stacking [73], and other approaches [77]. Interestingly, none of these studies have yet been applied to the extraction of food-related entities. Thus, we proposed a Recurrent Networks-based Ensemble or RNE model for extracting ingredient entities and their attributes in this study [19].

Table 4.1 Comparative overview of prior studies on NER utilizing the ensemble method

Study	**Ensemble Scheme**	**Classifiers**	**Dataset**	**Performance Metrics**
[71]	Voting	5 NER systems: Stanford NER, NER-Tagger, Edinburgh Geoparser, spaCy NER, Polyglot	The Marry Hamilton Papers [78], The Samuel Hartlib Papers [79]	Recall, Precision, F1 score
[69]	Voting	MLP, ABM1, J48	Reuters corpus [80]	Recall, Precision, Error rate, MCC, F1 score
[81]	Majority Voting	BERT, CNN, CamemBERT, CamemBERT-bio, XL-	ChEMU 2020 [82], DEFT 2020 [83], WNUT 2020 [84]	F1 score

Study	Ensemble Scheme	Classifiers	Dataset	Performance Metrics
		Net, RoBERTa, BioBERT, Bio + ClinicalBERT, PubMedBERT, BioMed RoBERTa		
[74]	Majority Voting	BERT-base-cased, BERT-based-uncased, CNN	ChEMU NER Task [82]	Recall, Precision, F1 score
[73]	Majority Voting, Stacking	SVM, CRF, ME	i2b2 2010 corpus [85]	Recall, Precision, F1 score
[70]	Weighted Voting	ME, CRF, SVM	Bengali News corpus [86], NERSSEAL [87], CoNLL-2003 [88]	F1 score
[75]	Plurality Voting, Weighted Voting	HMM, CRF, MEM, BiLSTM	Private (Authors private data)	F1 score
[77]	Arbitration Rules, Stacked Generalization, Cascade Generalization	Generalized Winnow, ME, SVM, CRF	GENIA [89], JNLPBA [90]	Recall, Precision, F1 score
[76]	Concatenation	Neural Networks	OKE2016 [91], AIDA/CoNLL [88], NexGenTV corpus [92]	Recall, Precision, F1 score

4.1.3 Recurrent Model for NER

Deep learning for NLP has resulted in a new research paradigm. a number of studies on deep learning-related models and methods have been applied to diverse NLP tasks [93]. Among these are recurrent neural networks. In modeling sequential

data, RNNs and their variations, such as GRU and LSTM, have demonstrated outstanding performance [94], [95]. Specifically, bidirectional RNNs efficiently utilize past information (via forward states) and future information (through backward states) for a given period [96]. The first work using RNNs for NER problems was conducted using the LSTM–CRF architecture by [96]. After this, several additional studies explored RNNs for sequence-labeling problems. In completing their NER tasks, the authors of [97]–[100] used GRU, whereas the authors of [101]–[104] used LSTM. In addition, the authors of [59] use RNNs for NER in Chinese electronic medical records and the authors of [105] employ RNNs for nested NER problems. In the case of food information extraction, BuTTER [15] and MenuNER [3] were developed using Bi-LSTM-CRF.

4.2 Dataset

In this study, we used the FINER dataset [18] we constructed in Chapter 3. The labels are spanned in the IOB chunking format of the data and have five kinds of named entity tags such as: INGREDIENT, PRODUCT, UNIT, QUANTITY, and STATE. The data set consists of 181,970 data or ingredient sentences, which are split 80:20 between training and testing, respectively. It is approximately 145,576 data for training and 36,394 data for testing the system. The complete distribution of the dataset across tags is further presented in Table 4.2.

Table 4.2 The distribution of the named entity dataset.

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

Named-entity type	Count	Ratio (%)
B-UNIT	174,993	12.52
I-INGREDIENT	240,436	17.20
I-PRODUCT	55,212	3.95
I- QUANTITY	1,919	0.14
I-STATE	130,158	9.31
I-UNIT	2,353	0.17
O (outside or non-entity chunk)	220,300	15.76
Total	**1,397,960**	**100**

4.3 Hyperparameter Optimization

As AI advances, deep learning techniques are becoming increasingly popular for any ML problems. However, the training process for these models involves a large number of hyperparameters, and the selection of these hyperparameters relies on experience. Thus, adjusting hyperparameters is a tedious and time-consuming process. This study uses the greedy search algorithm to find the best hyperparameter in search space or a dictionary where the hyperparameter arguments and values are used for the greedy hyperparameter search. In greedy search, the validation accuracy for each hyperparameter is determined locally. The greedy search algorithm maximizes each hyperparameter while keeping the others constant. In this strategy, the local optimum solution is obtained by optimizing the local solution for each hyperparameter using an iterative procedure that is repeated until all hyperparameters are optimal [106], [107]. Since model performance depends on the hyperparameters, all candidate classifiers are optimized prior to selection. The final values for each classifier based on the greedy hyperparameter optimization are presented in Table 4.3.

Table 4.3 Hyperparameter space for each model. The final value is determined by selecting the optimal hyperparameter using greedy search.

Model	Hyperparameter	Values	Final Value
RNN	*n_layers*	[1, 2, 4]	4
	hidden_dim	[16, 32, 64]	16
	dropout	[0.0, 0.3, 0.5]	0.0
GRU	*n_layers*	[1, 2, 4]	2
	hidden_dim	[16, 32, 64]	64
	dropout	[0.0, 0.3, 0.5]	0.3
LSTM	*n_layers*	[1, 2, 4]	2
	hidden_dim	[16, 32, 64]	32
	dropout	[0.0, 0.3, 0.5]	0.0

4.4 Recurrent Network-based Ensemble (RNE)

According to the wisdom of the crowd theory holds that collective knowledge is superior to that of the few [108]. In light of this, the ensemble method aims to improve prediction performance by aggregating the results of multiple models. A set of model predictions from the first-level classifiers are used as inputs for the second-level learning model in an ensemble approach. The second-level model is trained to optimally incorporate the predictions of the first-level classifiers to generate the final prediction. Thus, ensemble learning can transform a group of weak classifiers into a strong classifier [28]. It has been demonstrated that the ensemble method provides better accurate predictions than single models in a wide range of scenarios. When the models produce different results, the potential for performance improvement is higher when an ensemble method is used [109].

In this study, we develop a named-entity recognition model called the recurrent network-based ensemble (RNE) method to extract ingredient entities and their attributes from recipe text [19]. The RNE is comprised of recurrent network models such as RNN, GRU, and LSTM. It was developed utilizing a deep ensemble-learning framework. These models are independently trained on the same dataset and then integrated to create more accurate predictions for the extraction of food entities, such as ingredient names, products, units, quantities, and states for each ingredient in a recipe. The RNE model architecture is shown in Figure 4.1.

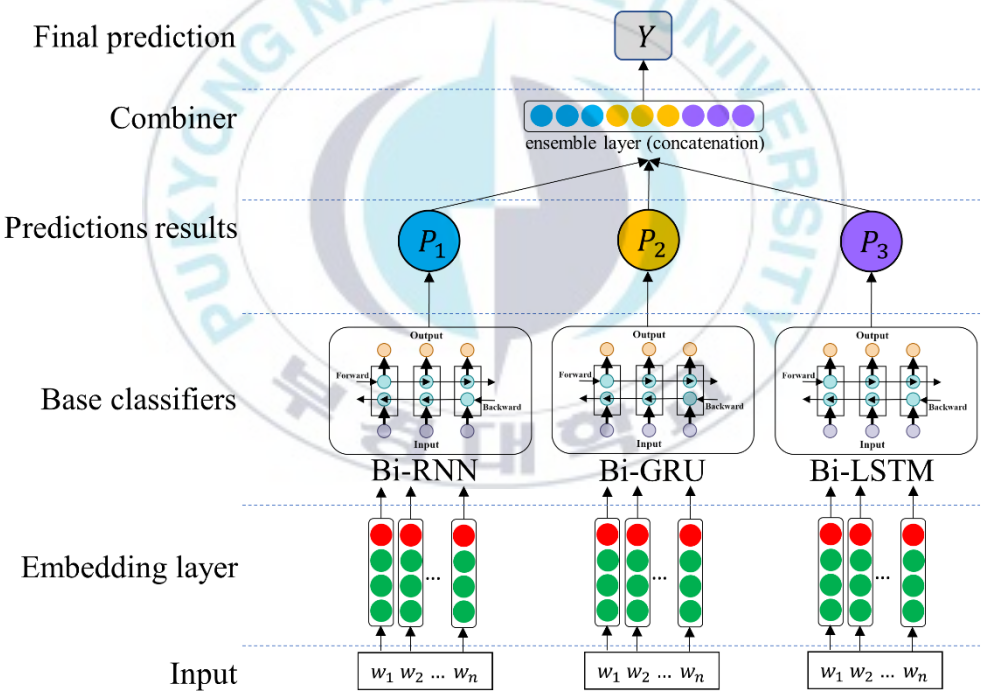


Figure 4.1 The RNE model architecture.

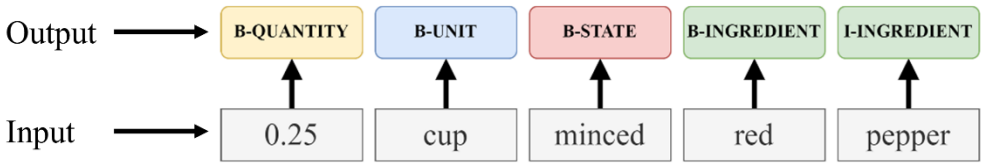


Figure 4.2 Input and output data example.

The input data to RNE is a list of ingredient phrases from the FINER dataset, named-entity dataset we have created in Chapter 3. In the first layer, we retrieved the contextual meaning of words using pre-trained GloVe [110] word embeddings. In this layer, the pre-trained word embedding is used to map each word ($\boldsymbol{w}_1, \boldsymbol{w}_2$, ... $\boldsymbol{w}_n$) in the sentence $\boldsymbol{X}$ to a word vector. In the next layer, we independently train three base classifiers 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 recurrent models, such as Bi-RNN, Bi-GRU, and Bi-LSTM, as the base classifiers. Each model generates 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 4.2. 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 $\circ$ representing the concatenation operator and $\boldsymbol{\mathcal{F}}: \mathbb{R}^{(D\times 3)} \rightarrow \mathbb{R}^K$ representing 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 4.2 illustrates an NER sentence input and output. All implementation details for the experiments conducted in this study are detailed in the experimental section.

Chapter V

Results and Analysis

5.1 Experimental Setup

All experiments are performed on a computer with an Intel(R) Core (TM) i9-10900KF processor running at 3.70 GHz, a NVidia GeForce RTX 3090 graphics processing unit, and 32 GB of RAM. And the Pytorch python package in an open-source Anaconda GPU environment with Python version 3.9.

5.2 Evaluation Metrics

To evaluate our model performance, we employ three different metrics [111]: Precision, Recall, and F1-score. We count the true positives (TP_t), false positives (FP_t), and false negatives (FN_t). TP_t occurs when the outputs of the NER for input tokens exactly matches the same ingredient entity in the ground truth dataset, FP_t or falsely predicted positive occurs when something that is not an ingredient entity is classified as being one, and FN_t occurs when a specific annotation is omitted when the entity should be classified as an ingredient entity and this happens when the ingredient entity is not properly extracted using the NER method. In addition, we use the one-against-all method to convert the multiclass

confusion matrix of each dataset into a binary confusion matrix [112]. These metrics for evaluation are calculated as follows:

$$\text{Precision} = \sum_{t \in T} \frac{TP_t}{(TP_t + FP_t)} \tag{20}$$

$$\text{Recall} = \sum_{t \in T} \frac{TP_t}{(TP_t + FN_t)} \tag{21}$$

$$\text{F1} = \sum_{t \in T} \frac{2 \times Precision_t Recall_t}{(Precision_t + Recall_t)} \tag{22}$$

where:

- **Precision** is the ratio between the number of true positives and the total number of predicted positives.
- **Recall** is the proportion of actual positives to true positives.
- **F1-Score** represents the harmonic mean of precision and recall.

5.3 Analysis and Evaluation for SMPT method

This section presents the analysis and then discuss the results of the experiments performed in this study 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, the dataset of this study can be accessed in Figshare [18] (URL: *https://doi.org/10.6084/m9.figshare.20222361.v3*).

5.3.1 Test Results with Training Schemes

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 the data quality and process efficiency. Figures 5.1, 5.2, and 5.3 gives a detailed picture about the performance of the three schemes with growth factor of 2, 5, and 10 respectively. These three figures show that the annotators' performances were increased as the dataset grows over iterations. This matches our expectation that the models improve with more training data, which is a strong indicator about the dataset quality.

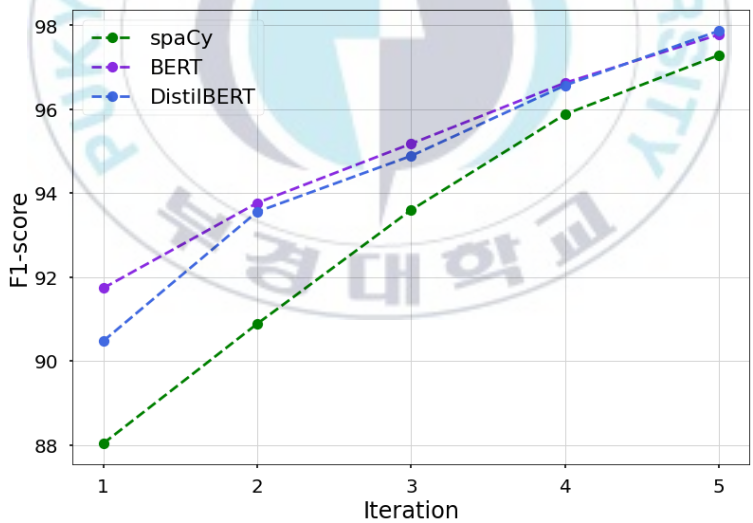


Figure 5.1 The performance results in each iteration with growth factor 2 (scheme 1).

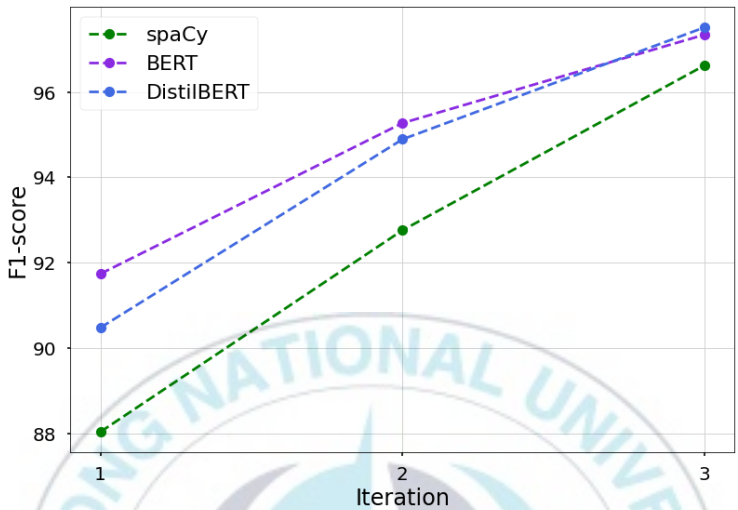


Figure 5.2 The performance results in each iteration with growth factor 5 (scheme 2).

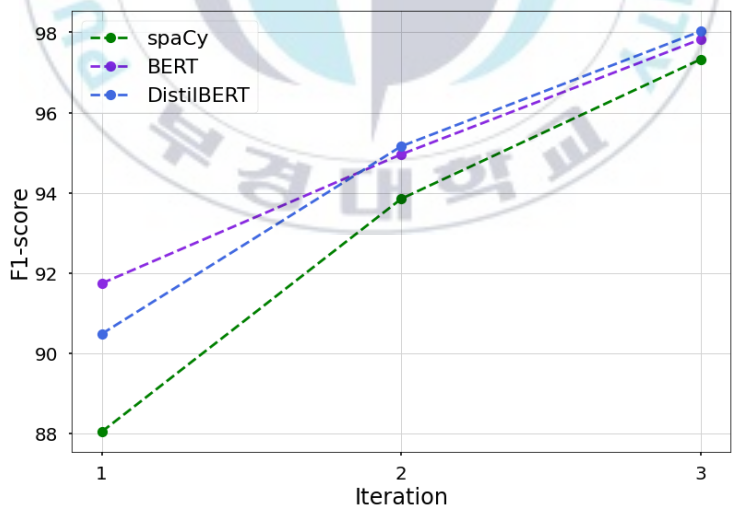


Figure 5.3 The performance results in each iteration with growth factor 10 (scheme 3).

Figure 5.4 presents the boxplot that compares the three schemes in terms of the average performance measured by the F1 score. The peak of the box represents the data point with the highest value, while the bottom shows the data point with the lowest value. A horizontal red line within the rectangle indicates all values' median, and the white diamond indicates the mean value. Figure 5.4 does not have outliers for all models. The length of the box plot indicates a more varied model distribution. According to the box plot, Scheme 3 outperforms all other models, with the distribution's median value being within the third quartile. It implies that 75% of the results are beneath the upper quartile.

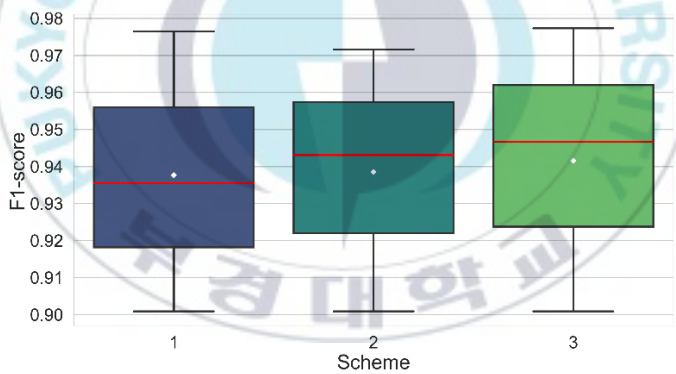


Figure 5.4 The average performance in each scheme.

In addition, Table 5.1 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 all schemes, according to Figure 5.4. 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.

Table 5.1 Computation time of each scheme. As the training set grows, labeling time increases. Note that the three schemes increased 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	2,000	146	5,000	268	10,000	914
2	6,000	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 5.5 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 boxes, and the correct prediction is highlighted in the blue boxes. For instance, the model returned a number of "*O*" tags for non-entity words. In the second iteration, the model began to learn, although, 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 5.5 An example of NER sentence generated in scheme 3 over three iterations.

5.3.2 Evaluation on ML 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 [113]–[115], BiLSTM-CRF [15], [101]–[103], [116], [117], and BERT [3], [66], [118]. They have proven to be effective for token classification tasks like NER [94], [95], [119].

The three models were trained on our FINER dataset, and we utilized the reserved evaluation set of 1,000 annotated samples, which we have prepared in Subchapter 3.3. Table 5.2 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. BERT's micro-average yields precision, recall, and F1-score of 0.978, 0.980, and 0.979, respectively. Its macro-average metrics were 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 simply 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 5.2 Performance of each model with the best performance is emphasized in **bold**.

Evaluation metrics	**CRF**		**Bi-LSTM-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**

Table 5.3 The classification report for each models and the best performance is emphasized in **bold**.

Class	**CRF**			**Bi-LSTM-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**	**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**

Class	CRF			Bi-LSTM-CRF			BERT		
	Precision	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
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	1	1	0.999	0.998	0.999	**1**	**1**	**1**

Additionally, the CRF model used in this experiment achieved the scores listed in Tables 5.2 and Table 5.3 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. Compared to them, the FINER dataset outperformed both performances by an F1 score of 97.9%, as shown in Table 5.4.

Table 5.4 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 [8], [13]	*K*-Means Clustering	0.961
TASTEset [11]	BERT	0.935
FINER (Our)	**BERT**	**0.979**

For further analysis and verification, we presented a detailed evaluation of each entity tag of these models in Table 5.3. 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 probability of occurring promptly after QUANTITY, making it relatively easy for the model to predict them. Overall, BERT performed better than other models in most cases. The BERT has been pre-trained on a large corpus and it preserves powerful representations of the language, so it is not surprising that it performs best in most downstream tasks like NER. Based on these experiments, we conclude that our method is helpful and beneficial in building a reasonably good data set for typical NER tasks

5.4 Analysis and Evaluation for RNE model

The proposed model is evaluated using evaluation metrics including precision, recall, and F1-score. Using these metrics, we analyzed and compared the proposed Recurrent Network-based Ensemble method (RNE) model with the single models of RNN, GRU, and LSTM. Table 5.5 provides a summary of the performance of both unidirectional and bidirectional models. A unidirectional or one-way network stores only forward information (past to future). While the bidirectional approach utilizes information from both sides and accepts input flows in both directions (past to future and vice versa).

Table 5.5 A comparison of the models of unidirectional and bidirectional recurrent networks. The best performance is emphasized in **bold**.

Model	**F1-score (%)**	
	Unidirectional	**Bidirectional**
RNN	94.26	95.60
GRU	**94.86**	95.83
LSTM	94.18	95.59
RNE	94.83	**96.00**

According to Table 5.5, we observe that that RNE with a bidirectional network type achieved the highest performance of 96%, outperformed all single models by 0.2-0.4% in F1-score. The results indicate that the bidirectional network type is superior to the unidirectional network type. This architecture has several advantages for solving real-world issues, particularly in NLP. Each component of the input sequence contains past and present information. Because the bidirectional type combines network layers from both directions, it can produce more relevant output. In contrast, RNN performed the worst of all models, which was to be expected considering that GRU, LSTM, and RNE have more complex architectures than RNN. We believe these findings suggest that GRU and LSTM are superior to RNN due to the use of memory cells with a gate mechanism for controlling the flow of information in sequential data [94].

In addition, we analyzed and compared the computing cost of each model in Figure 5.6. The figure demonstrates the average training time for four models on a computer with an Intel(R) Core (TM) i9-10900KF

processor running at 3.70 GHz, a NVidia GeForce RTX 3090 graphics processing unit, and 32 GB of RAM. The Bi-LSTM and Bi-RNN model outperformed all other models in terms of time efficiency. Both models performed, on average, 36.14% % faster than the second-fastest model, Bi-GRU, and 143,4% faster than RNE. Thus, it makes RNE is 239.6% slower than both of Bi-RNN and Bi-LSTM. Due to the expertise and time required to train and maintain several models instead of a single model, the RNE model is the most time-consuming as.

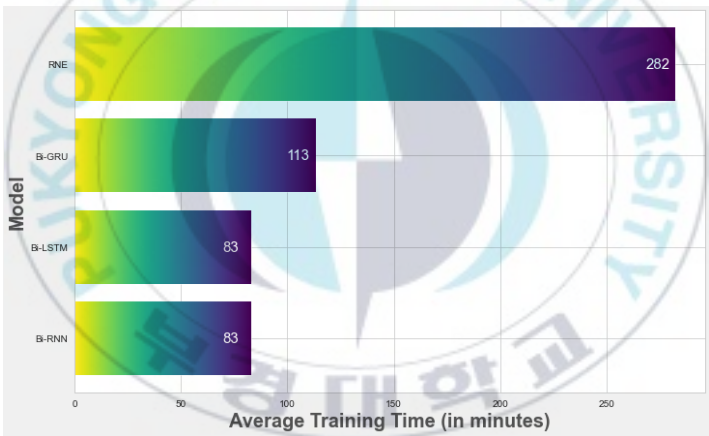


Figure 5.6 The computational cost comparison for each model.

Table 5.6 presents detailed performance for each entity for the bidirectional type of RNN, GRU, LSTM, and RNE models. Model performance ranges from 89% for B-PRODUCT and 100% for I-UNIT. For inside-tag UNIT or I-UNIT, most models attain perfect performance, and this is because I-UNITs are very rare (account for 0.17% of the overall dataset), and the majority of them correspond to the *"fluid ounces"* token,

making prediction relatively easy for the model. The same holds true for the performance of B-STATE, which has nearly 100 percent accuracy across all models, despite the fact that the number of entities in the dataset is around 9.7%, they have a small number of unique tokens, making it relatively easy to predict. In contrary, the B-PRODUCT entity has the lowest performance, not only due to its limited annotation distribution in the data set (about 1.2%) but also due to the difficulty in detecting the beginning tag of the PRODUCT. The PRODUCT entity is often long, unique, and scarce. It consists of many tokens, as shown in an example of product entity extraction in Figure 3.2 on the previous Subchapter 3.3. Therefore, it appears that the amount of training data samples for this entity is insufficient for automatic entity learning.

Figures 5.7, 5.8, 5.9, and 5.10 demonstrate that across all models, most misclassifications were due to incorrect predictions of both the beginning and inside chunk of the class INGREDIENT, PRODUCT, and QUANTITY against all entities, except the class I-UNIT. To improve the model's performance, one should focus on the predictive results of these three classes, which is the highest misclassification rate among all the classes. In contrast, class I-UNIT has zero misclassifications across all entity models. Overall, the performance findings presented in Table 5.5, Table 5.6, and the confusion matrix in Figures 5.7, 5.8, 5.9, and 5.10 show that RNE can be the method of choice for NER. It can be seen that compared to the single models, our proposed model enhances the detection of ingredient entities by 0.2 to 0.4%.

Table 5.6 The classification report for each NER model with their highest performance emphasized in **bold**. **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	0.962	0.970	0.966	0.962	0.973	0.968	0.954	**0.975**	0.964	**0.968**	0.971	**0.970**
B-PRODUCT	0.890	0.922	0.910	**0.910**	0.927	**0.918**	0.902	0.918	0.911	0.899	**0.935**	0.916
B- QUANTITY	**0.967**	0.974	**0.971**	0.950	**0.978**	0.963	0.959	0.976	0.967	0.954	**0.978**	0.966
B-STATE	**0.999**	0.998	**0.999**	**0.999**	**0.999**	**0.999**	**0.999**	0.998	**0.999**	**0.999**	**0.999**	**0.999**
B-UNIT	**0.997**	0.997	**0.997**	0.995	**0.998**	**0.997**	0.996	**0.998**	**0.997**	**0.997**	**0.998**	**0.997**
I-INGREDIENT	0.966	0.970	0.968	0.965	**0.978**	0.971	0.966	0.972	0.969	**0.968**	0.976	**0.972**
I-PRODUCT	0.925	**0.976**	0.950	0.950	0.970	**0.960**	0.941	0.968	0.954	**0.951**	0.970	**0.960**
I-QUANTITY	0.926	0.968	0.946	0.920	0.971	0.945	**0.929**	0.970	0.944	0.925	**0.974**	**0.949**
I-STATE	**0.937**	0.964	**0.951**	0.917	0.967	0.941	0.881	0.968	0.922	0.925	**0.970**	0.947
I-UNIT	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

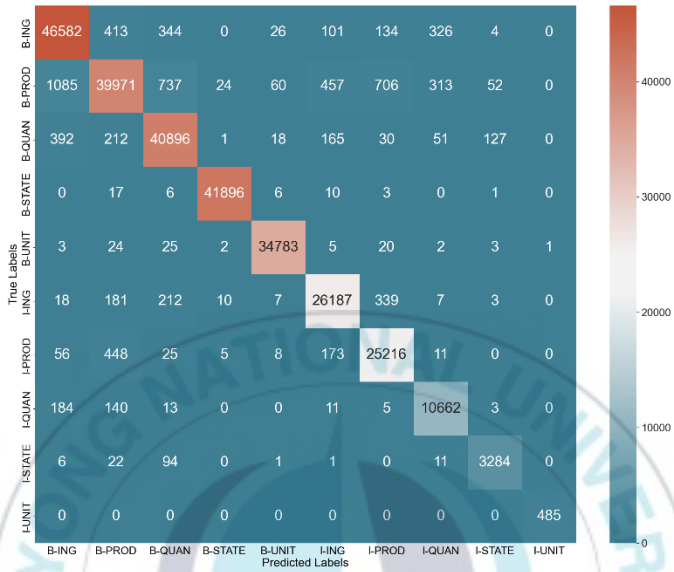


Figure 5.7 The confusion matrix analysis for Bi-RNN model.

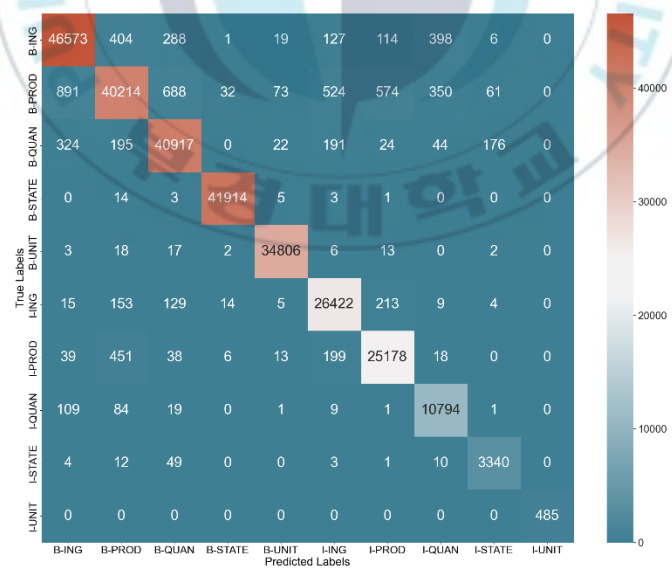


Figure 5.8 The confusion matrix analysis for Bi-GRU model.

Figure 5.9 The confusion matrix analysis for Bi-LSTM model.

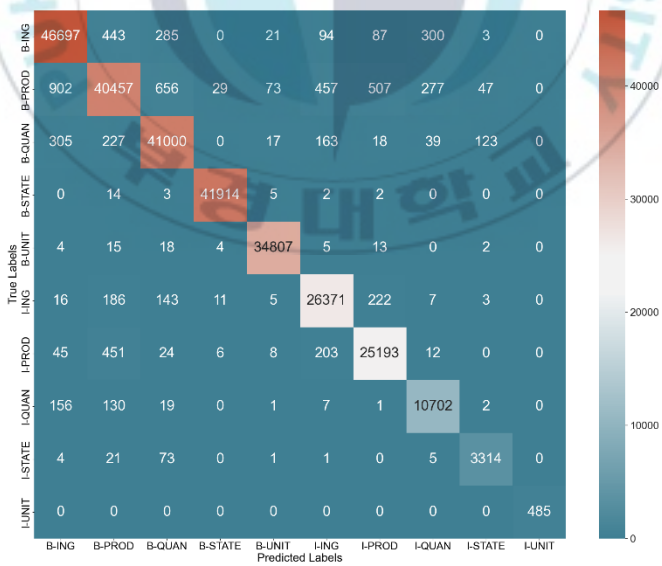


Figure 5.10 The confusion matrix analysis for RNE model.

However, the computational cost illustrated in Figure 5.6 implies that, given the model's simplicity and computational time, a single model such as Bi-RNN or Bi-LSTM could be the second-best alternative after RNE. Although Bi-GRU performs marginally better, Bi-RNN and Bi-LSTM are faster at training and computationally less expensive than Bi-GRU and RNE. In addition, it is important to note that there is no guarantee that the combination of several classifiers will always exceed the ensemble's best individual classifier, with the exception of certain cases [120]. Therefore, combining classifiers may not always outperform the best classifier in the ensemble, but it reduces the likelihood of making a very poor selection.

Chapter VI
Conclusions and Future Work

6.1 Conclusions

To address the limited resources available for food information extraction, particularly for NER tasks. We construct a named entity recognition dataset for food ingredients. In constructing the dataset, we introduced a method based on self-training and ensemble learning called SMPT (Semi-supervised Multi-Model Prediction Technique). SMPT is a semi-supervised method for building a NER dataset iteratively by incorporating the concept of self-training applied to pretrained models such as spaCy NER, BERT, and DistilBERT under a voting scheme mechanism for improved prediction. After a series 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 [18].

Furthermore, in order to enhance the performance of NER model in extracting ingredient entities from recipe text, a recurrent network-based ensemble model (RNE) was proposed. For the final prediction, we aggregated the predicted results of bidirectional RNN, GRU, and LSTM using the ensemble concatenation approach for the final prediction. The evaluation findings indicate that the ensemble model performs better than a single model. Due to the reduction of the variance component of the prediction error made by the contributing model, the ensemble model achieves a higher performance. Thus, it can be said that the ensemble method can enhance the classifier's performance and reduce variance.

6.2 Future Work

As we highlighted in section 5.4 regarding the efficiency of the RNE model, a single model such as Bi-RNN and Bi-LSTM can be the second option after RNE due to its simplicity and computational time. Although Bi-GRU performs slightly better, Bi-RNN and Bi-LSTM are faster at training and less expensive computationally than other models. Furthermore, it may be possible in the future to improve these values and design a more robust model that maintains a balance between computational cost and model performance by incorporating additional factors such as exploring better architectures, increasing the size of the dataset, and selecting a more appropriate hyperparameters setting. As a result, we offer the following suggestions for future research.

(1) In the future this model can be leverage for cross-domain adaptation with adding one or more layer before the base classifiers layer. The new layer can be a neural adaptation layer using domain specific pre-trained word embeddings coupled with character level CNN or LSTM.

(2) For better understanding of the words and its context in a particular sentence, in word embeddings layer part instead of using pre-trained word embedding like Word2vec and GloVE, we can utilize ELMo or BERT for feature extraction. Word2vec and GloVE word embeddings are context independent. Thus, these models output just one vector (embedding) for each word, combining all the different senses of the word into one vector. On the other hand, ELMo and BERT is context dependent which can generate different word embeddings for a word that captures the context of a word based on its position in a sentence.

(3) In the final layer, instead of using concatenation we can employed stacking ensemble with CRF layer as the final layer. In stacking ensemble, we take the

probability from the 3 models and using it as an input to the second layer. In the second layer we can utilize meta-classifier such as MLR (multinomial linear regression) or RF (random forest). And then in the final layer for better prediction we add CRF layer. Other than stacking, we can employ other ensemble learning approach to enhance the model performance and effectiveness in detecting entities.

(4) For future development, we can further enlarge the dataset by incorporating additional information from multiple food-sharing websites such as food.com, yummly.com, cookpad.com, and many more.

(5) In the future, we can use the FINER dataset to develop chatbot systems for food-related applications, such as a personalized diet Q&A chatbot. In addition, we can integrate this data set to a personalized food recommendation system to calculate the nutrition intake.

References

[1] W. Min, C. Liu, L. Xu, and S. Jiang, "Applications of knowledge graphs for food science and industry," *Patterns*, vol. 3, no. 5, p. 100484, May 2022, doi: 10.1016/j.patter.2022.100484.

[2] V. Krishnan and V. Ganapathy, "Named Entity Recognition," 2005. http://cs229.stanford.edu/proj2005/KrishnanGan apathy-NamedEntityRecognition.pdf. (accessed Feb. 04, 2021).

[3] M. H. Syed and S.-T. Chung, "MenuNER: Domain-Adapted BERT Based NER Approach for a Domain with Limited Dataset and Its Application to Food Menu Domain," *Applied Sciences*, vol. 11, no. 13, p. 6007, Jun. 2021, doi: 10.3390/app11136007.

[4] Kokoy Siti Komariah and Bong-Kee Sin, "Nutrition-Based Food Recommendation System for Prediabetic Person," in *Korea Software Congress 2020*, Pyeongchang, Rep. of Korea, 2021.

[5] C. Pellegrini, E. Özsoy, M. Wintergerst, and G. Groh, "Exploiting Food Embeddings for Ingredient Substitution," in *HEALTHINF*, 2021.

[6] J. Kalra, D. Batra, N. Diwan, and G. Bagler, "Nutritional Profile Estimation in Cooking Recipes," in *2020 IEEE 36th International Conference on Data Engineering Workshops (ICDEW)*, Dallas, TX, USA, Apr. 2020, pp. 82–87. doi: 10.1109/ICDEW49219.2020.000-3.

[7] G. Popovski, B. K. Seljak, and T. Eftimov, "FoodBase corpus: a new resource of annotated food entities," *Database*, vol. 2019, p. baz121, Jan. 2019, doi: 10.1093/database/baz121.

[8] D. Batra *et al.*, "RecipeDB: a resource for exploring recipes," *Database*, vol. 2020, p. baaa077, Nov. 2020, doi: 10.1093/database/baaa077.

[9] M. Bień, M. Gilski, M. Maciejewska, W. Taisner, D. Wisniewski, and A. Lawrynowicz, "RecipeNLG: A Cooking Recipes Dataset for Semi-Structured Text Generation," in *Proceedings of the 13th International Conference on Natural Language Generation*, Dublin, Ireland, Dec. 2020, pp. 22–28. [Online]. Available: https://aclanthology.org/2020.inlg-1.4

[10] J. Marin *et al.*, "Recipe1M+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images," 2018, doi: 10.48550/ARXIV.1810.06553.

[11] A. Wróblewska, A. Kaliska, M. Pawłowski, D. Wiśniewski, W. Sosnowski, and A. Ławrynowicz, "TASTEset -- Recipe Dataset and Food Entities Recognition Benchmark," 2022, doi: 10.48550/ARXIV.2204.07775.

[12] G. Popovski, B. K. Seljak, and T. Eftimov, “A Survey of Named-Entity Recognition Methods for Food Information Extraction,” *IEEE Access*, vol. 8, pp. 31586–31594, 2020, doi: 10.1109/ACCESS.2020.2973502.

[13] N. Diwan, D. Batra, and G. Bagler, “A Named Entity Based Approach to Model Recipes,” in *2020 IEEE 36th International Conference on Data Engineering Workshops (ICDEW)*, 2020, pp. 88–93. doi: 10.1109/ICDEW49219.2020.000-2.

[14] G. Popovski, S. Kochev, B. Seljak, and T. Eftimov, “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*, Prague, Czech Republic, 2019, pp. 915–922. doi: 10.5220/0007686309150922.

[15] G. Cenikj, G. Popovski, R. Stojanov, B. K. Seljak, and T. Eftimov, “BuTTER: BidirecTional LSTM for Food Named-Entity Recognition,” in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 3550–3556. doi: 10.1109/BigData50022.2020.9378151.

[16] R. Stojanov, G. Popovski, G. Cenikj, B. Koroušić Seljak, and T. Eftimov, “A Fine-Tuned Bidirectional Encoder Representations From Transformers Model for Food Named-Entity Recognition: Algorithm Development and Validation,” *J Med Internet Res*, vol. 23, no. 8, p. e28229, Aug. 2021, doi: 10.2196/28229.

[17] W. Min, S. Jiang, L. Liu, Y. Rui, and R. Jain, “A Survey on Food Computing,” *ACM Comput. Surv.*, vol. 52, no. 5, pp. 1–36, Sep. 2020, doi: 10.1145/3329168.

[18] K. Siti Komariah, B.-K. Sin, and A. Tulus Purnomo, “FINER: Food Ingredient NER Dataset.” Apr. 07, 2022. Accessed: May 09, 2022. [Online]. Available: https://doi.org/10.6084/m9.figshare.20222361.v3

[19] K. S. Komariah and B.-K. Sin, “Enhancing Food Ingredient Named-Entity Recognition with Recurrent Network-Based Ensemble (RNE) Model,” *Applied Sciences*, vol. 12, no. 20, p. 10310, Oct. 2022, doi: 10.3390/app122010310.

[20] “Natural language processing.” Accessed: Oct. 12, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Natural_language_processing

[21] D. Khurana, A. Koli, K. Khatter, and S. Singh, “Natural language processing: state of the art, current trends and challenges,” *Multimed Tools Appl*, Jul. 2022, doi: 10.1007/s11042-022-13428-4.

[22] Sowmya V. B, B. Majumder, A. Gupta, and H. Surana, *Practical natural language processing: a comprehensive guide to building real-world NLP systems*, First edition. Sebastopol, CA: O’Reilly Media, 2020.

[23] J. Saunders and T. Smith, “Malnutrition: causes and consequences,” *Clin Med*, vol. 10, no. 6, pp. 624–627, Dec. 2010, doi: 10.7861/clinmedicine.10-6-624.
[24] U. Leser and J. Hakenberg, “What makes a gene name? Named entity recognition in the biomedical literature,” *Briefings in Bioinformatics*, vol. 6, no. 4, pp. 357–369, Jan. 2005, doi: 10.1093/bib/6.4.357.
[25] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. Boca Raton, FL: Taylor & Francis, 2012.
[26] R. Maclin and D. W. Opitz, “Popular Ensemble Methods: An Empirical Study,” *CoRR*, vol. abs/1106.0257, 2011, [Online]. Available: http://arxiv.org/abs/1106.0257
[27] T. G. Dietterich, “Ensemble Methods in Machine Learning,” in *Multiple Classifier Systems*, 2000.
[28] C. C. Aggarwal, Ed., *Data classification: algorithms and applications*. Boca Raton: CRC Press, Taylor & Francis Group, 2014.
[29] D. Sarkar and V. Natarajan, *Ensemble machine learning cookbook: over 35 practical recipes to explore ensemble machine learning techniques using Python*. Birmingham Mumbai: Packt, 2019.
[30] H. Abdel-Jaber, D. Devassy, A. Al Salam, L. Hidaytallah, and M. EL-Amir, “A Review of Deep Learning Algorithms and Their Applications in Healthcare,” *Algorithms*, vol. 15, no. 2, p. 71, Feb. 2022, doi: 10.3390/a15020071.
[31] S. Ali, K. Masood, A. Riaz, and A. Saud, “Named Entity Recognition using Deep Learning: A Review,” in *2022 International Conference on Business Analytics for Technology and Security (ICBATS)*, Dubai, United Arab Emirates, Feb. 2022, pp. 1–7. doi: 10.1109/ICBATS54253.2022.9759051.
[32] M. A. Ganaie, M. Hu, A. K. Malik, M. Tanveer, and P. N. Suganthan, “Ensemble deep learning: A review,” *Engineering Applications of Artificial Intelligence*, vol. 115, p. 105151, Oct. 2022, doi: 10.1016/j.engappai.2022.105151.
[33] L. K. Smirani, H. A. Yamani, L. J. Menzli, and J. A. Boulahia, “Using Ensemble Learning Algorithms to Predict Student Failure and Enabling Customized Educational Paths,” *Scientific Programming*, vol. 2022, pp. 1–15, Apr. 2022, doi: 10.1155/2022/3805235.
[34] Y. Cao, T. A. Geddes, J. Y. H. Yang, and P. Yang, “Ensemble deep learning in bioinformatics,” *Nat Mach Intell*, vol. 2, no. 9, pp. 500–508, Aug. 2020, doi: 10.1038/s42256-020-0217-y.
[35] V. Akpokiro, T. Martin, and O. Oluwadare, “EnsembleSplice: ensemble deep learning model for splice site prediction,” *BMC Bioinformatics*, vol. 23, no. 1, p. 413, Oct. 2022, doi: 10.1186/s12859-022-04971-w.

[36] O. A. Malik, M. Faisal, and B. R. Hussein, “Ensemble Deep Learning Models for Fine-grained Plant Species Identification,” in *2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, Brisbane, Australia, Dec. 2021, pp. 1–6. doi: 10.1109/CSDE53843.2021.9718387.

[37] H. Saleh, S. Mostafa, A. Alharbi, S. El-Sappagh, and T. Alkhalifah, “Heterogeneous Ensemble Deep Learning Model for Enhanced Arabic Sentiment Analysis,” *Sensors*, vol. 22, no. 10, p. 3707, May 2022, doi: 10.3390/s22103707.

[38] C. Becker *et al.*, *Modern Approaches in Natural Language Processing*. Department of Statistics, LMU Munich, 2020. Accessed: Oct. 19, 2022. [Online]. Available: https://slds-lmu.github.io/seminar_nlp_ss20/index.html

[39] S. Ruder, “Neural Transfer Learning for Natural Language Processing,” National University of Ireland, Galway, Ireland, 2019.

[40] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” 2018, doi: 10.48550/ARXIV.1810.04805.

[41] S. J. Pan and Q. Yang, “A Survey on Transfer Learning,” *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.

[42] M.-R. Amini, V. Feofanov, L. Pauletto, E. Devijver, and Y. Maximov, “Self-Training: A Survey,” 2022, doi: 10.48550/ARXIV.2202.12040.

[43] S. Ruder and B. Plank, “Strong Baselines for Neural Semi-supervised Learning under Domain Shift,” 2018, doi: 10.48550/ARXIV.1804.09530.

[44] A. Vaswani *et al.*, “Attention Is All You Need,” 2017, doi: 10.48550/ARXIV.1706.03762.

[45] J. Thickstun, “The Transformer Model in Equations.” Accessed: Mar. 15, 2022. [Online]. Available: https://johnthickstun.com/docs/transformers.pdf

[46] A. Radford and K. Narasimhan, “Improving Language Understanding by Generative Pre-Training,” 2018.

[47] M. Honnibal and I. Montani, “spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing,” 2017. [Online]. Available: https://spacy.io

[48] E. Partalidou, E. Spyromitros-Xioufis, S. Doropoulos, S. Vologiannidis, and K. I. Diamantaras, “Design and implementation of an open source Greek POS Tagger and Entity Recognizer using spaCy,” 2019, doi: 10.48550/ARXIV.1912.10162.

[49] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter,” 2019, doi: 10.48550/ARXIV.1910.01108.

[50] M. Hagiwara, *Real-world natural language processing*. Shelter Island: Manning Publications, 2022.
[51] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," 2015, doi: 10.48550/ARXIV.1503.02531.
[52] "Understanding LSTM Networks -- colah's blog." http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (accessed May 03, 2022).
[53] R. J. Williams and J. Peng, "An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories," *Neural Computation*, vol. 2, no. 4, pp. 490–501, Dec. 1990, doi: 10.1162/neco.1990.2.4.490.
[54] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994, doi: 10.1109/72.279181.
[55] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling." arXiv, Dec. 11, 2014. Accessed: Aug. 15, 2022. [Online]. Available: http://arxiv.org/abs/1412.3555
[56] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, Doha, Qatar, 2014, pp. 103–111. doi: 10.3115/v1/W14-4012.
[57] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
[58] Y. Deng, Y. Jiao, and B.-L. Lu, "Driver Sleepiness Detection Using LSTM Neural Network," in *Neural Information Processing*, vol. 11304, L. Cheng, A. C. S. Leung, and S. Ozawa, Eds. Cham: Springer International Publishing, 2018, pp. 622–633. doi: 10.1007/978-3-030-04212-7_55.
[59] S. Chowdhury *et al.*, "A multitask bi-directional RNN model for named entity recognition on Chinese electronic medical records," *BMC Bioinformatics*, vol. 19, no. S17, p. 499, Dec. 2018, doi: 10.1186/s12859-018-2467-9.
[60] K. Cho *et al.*, "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)*, Doha, Qatar, 2014, pp. 1724–1734. doi: 10.3115/v1/D14-1179.
[61] S. S. Boushehri, A. B. Qasim, D. Waibel, F. Schmich, and C. Marr, "Annotation-efficient classification combining active learning, pre-training and semi-supervised learning for biomedical images," *bioRxiv*, p. 2020.12.07.414235, Jan. 2020, doi: 10.1101/2020.12.07.414235.

[62] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009. [Online]. Available: https://www.nltk.org
[63] H. Nakayama, T. Kubo, J. Kamura, Y. Taniguchi, and X. Liang, "doccano: Text Annotation Tool for Human." 2018. [Online]. Available: https://github.com/doccano/doccano
[64] "Allrecipes." https://www.allrecipes.com/ (accessed Sep. 10, 2021).
[65] L. A. Ramshaw and M. P. Marcus, "Text Chunking Using Transformation-Based Learning," in *Natural Language Processing Using Very Large Corpora*, vol. 11, S. Armstrong, K. Church, P. Isabelle, S. Manzi, E. Tzoukermann, and D. Yarowsky, Eds. Dordrecht: Springer Netherlands, 1999, pp. 157–176. doi: 10.1007/978-94-017-2390-9_10.
[66] J. Kim, Y. Ko, and J. Seo, "Construction of Machine-Labeled Data for Improving Named Entity Recognition by Transfer Learning," *IEEE Access*, vol. 8, pp. 59684–59693, 2020, doi: 10.1109/ACCESS.2020.2981361.
[67] Y. Wen, C. Fan, G. Chen, X. Chen, and M. Chen, "A Survey on Named Entity Recognition," in *Communications, Signal Processing, and Systems*, vol. 571, Q. Liang, W. Wang, X. Liu, Z. Na, M. Jia, and B. Zhang, Eds. Singapore: Springer Singapore, 2020, pp. 1803–1810. doi: 10.1007/978-981-13-9409-6_218.
[68] T. Eftimov, B. Koroušić Seljak, and P. Korošec, "A rule-based named-entity recognition method for knowledge extraction of evidence-based dietary recommendations," *PLoS ONE*, vol. 12, no. 6, p. e0179488, Jun. 2017, doi: 10.1371/journal.pone.0179488.
[69] R. Speck and A.-C. Ngonga Ngomo, "Ensemble Learning for Named Entity Recognition," in *The Semantic Web – ISWC 2014*, vol. 8796, P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. Knoblock, D. Vrandečić, P. Groth, N. Noy, K. Janowicz, and C. Goble, Eds. Cham: Springer International Publishing, 2014, pp. 519–534. doi: 10.1007/978-3-319-11964-9_33.
[70] A. Ekbal and S. Saha, "Weighted Vote-Based Classifier Ensemble for Named Entity Recognition: A Genetic Algorithm-Based Approach," *ACM Transactions on Asian Language Information Processing*, vol. 10, no. 2, pp. 1–37, Jun. 2011, doi: 10.1145/1967293.1967296.
[71] M. Won, P. Murrieta-Flores, and B. Martins, "Ensemble Named Entity Recognition (NER): Evaluating NER Tools in the Identification of Place Names in Historical Corpora," *Front. Digit. Humanit.*, vol. 5, p. 2, Mar. 2018, doi: 10.3389/fdigh.2018.00002.
[72] N. Naderi, J. Knafou, J. Copara, P. Ruch, and D. Teodoro, "Ensemble of Deep Masked Language Models for Effective Named Entity Recognition in Health

and Life Science Corpora," *Front. Res. Metr. Anal.*, vol. 6, p. 689803, Nov. 2021, doi: 10.3389/frma.2021.689803.

[73] H. Nayel and H. L. Shashirekha, "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, Dec. 2017, pp. 197–204. [Online]. Available: https://aclanthology.org/W17-7525

[74] J. Copara, N. Naderi, J. Knafou, P. Ruch, and D. Teodoro, "Named Entity Recognition in Chemical Patents using Ensemble of Contextual Language Models," *ArXiv*, vol. abs/2007.12569, 2020.

[75] Z. Jiang, "The Application of Ensemble Learning on Named Entity Recognition for Legal Knowledgebase of Properties Involved in Criminal Cases," in *2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications(AEECA)*, Dalian, China, Aug. 2020, pp. 701–705. doi: 10.1109/AEECA49918.2020.9213660.

[76] L. Canale, P. Lisena, and R. Troncy, "A Novel Ensemble Method for Named Entity Recognition and Disambiguation Based on Neural Network," in *The Semantic Web – ISWC 2018*, vol. 11136, D. Vrandečić, K. Bontcheva, M. C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L.-A. Kaffee, and E. Simperl, Eds. Cham: Springer International Publishing, 2018, pp. 91–107. doi: 10.1007/978-3-030-00671-6_6.

[77] H. Wang, T. Zhao, H. Tan, and S. Zhang, "Biomedical Named Entity Recognition Based on Classifiers Ensemble," *Int. J. Comput. Sci. Appl.*, vol. 5, pp. 1–11, 2008.

[78] A.-C. Gardner, M. Hundt, and M. Kindlimann, "Digitization of the Mary Hamilton Papers," *ICAME Journal*, vol. 41, no. 1, pp. 83–110, Mar. 2017, doi: 10.1515/icame-2017-0004.

[79] D. R Dickson, "The Hartlib Papers: A Complete Text and Image Database of the Papers of Samuel Hartlib (ca. 1600-1662)," *Isis*, vol. 88, no. 3, pp. 536–536, Sep. 1997, doi: 10.1086/383797.

[80] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "RCV1: A New Benchmark Collection for Text Categorization Research," *Journal of Machine Learning Research*, vol. 5, pp. 361–397, 2004.

[81] N. Naderi, J. Knafou, J. Copara, P. Ruch, and D. Teodoro, "Ensemble of deep masked language models for effective named entity recognition in multi-domain corpora," Health Informatics, preprint, Apr. 2021. doi: 10.1101/2021.04.26.21256038.

[82] J. He *et al.*, "ChEMU 2020: Natural Language Processing Methods Are Effective for Information Extraction From Chemical Patents," *Front. Res. Metr. Anal.*, vol. 6, p. 654438, Mar. 2021, doi: 10.3389/frma.2021.654438.

[83] S. Spala, N. A. Miller, F. Dernoncourt, and C. Dockhorn, “SemEval-2020 Task 6: Definition extraction from free text with the DEFT corpus,” in *SemEval-2020 Task 6: Definition extraction from free text with the DEFT corpus*, Florence, Italy, Aug. 2019, pp. 124–131. [Online]. Available: https://www.aclweb.org/anthology/W19-4015

[84] “2020 The 6th Workshop on Noisy User-generated Text (W-NUT).” [Online]. Available: http://noisy-text.github.io/2020/wlp-task.html

[85] “Fourth i2b2/VA Shared-Task and Workshop Challenges in Natural Language Processing for Clinical Data.” [Online]. Available: https://www.i2b2.org/NLP/Relations/

[86] A. Ekbal and S. Bandyopadhyay, “A web-based Bengali news corpus for named entity recognition,” *Lang Resources & Evaluation*, vol. 42, no. 2, pp. 173–182, May 2008, doi: 10.1007/s10579-008-9064-x.

[87] “IJCNLP-08 NERSSEAL shared task.” [Online]. Available: http://ltrc.iiit.ac.in/ner-ssea-08/

[88] E. F. T. K. Sang and F. De Meulder, “Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition,” 2003, doi: 10.48550/ARXIV.CS/0306050.

[89] J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii, “GENIA corpus--a semantically annotated corpus for bio-textmining,” *Bioinformatics*, vol. 19, no. Suppl 1, pp. i180–i182, Jul. 2003, doi: 10.1093/bioinformatics/btg1023.

[90] M.-S. Huang, P.-T. Lai, P.-Y. Lin, Y.-T. You, R. T.-H. Tsai, and W.-L. Hsu, “Revised JNLPBA Corpus: A Revised Version of Biomedical NER Corpus for Relation Extraction Task,” *Briefings in Bioinformatics*, vol. 21, no. 6, pp. 2219–2238, Dec. 2020, doi: 10.1093/bib/bbaa054.

[91] F. Nooralahzadeh, C. Lopez, E. Cabrio, F. Gandon, and F. Segond, “Adapting Semantic Spreading Activation to Entity Linking in Text,” in *Natural Language Processing and Information Systems*, vol. 9612, E. Métais, F. Meziane, M. Saraee, V. Sugumaran, and S. Vadera, Eds. Cham: Springer International Publishing, 2016, pp. 74–90. doi: 10.1007/978-3-319-41754-7_7.

[92] P. Torino, L. Farinetti, and R. Troncy, “NERD for NexGenTV Ensemble Learning for Named Entity Recognition and Disambiguation,” 2018. Accessed: Dec. 20, 2022. [Online]. Available: https://www.semanticscholar.org/paper/NERD-for-NexGenTV-Ensemble-Learning-for-Named-and-Torino-Farinetti/757cbfcb4a57c84ad09830bf9753269241793199

[93] T. Young, D. Hazarika, S. Poria, and E. Cambria, “Recent Trends in Deep Learning Based Natural Language Processing,” *CoRR*, vol. abs/1708.02709, 2017, [Online]. Available: http://arxiv.org/abs/1708.02709

[94] J. Li, A. Sun, J. Han, and C. Li, "A Survey on Deep Learning for Named Entity Recognition," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 50–70, 2022, doi: 10.1109/TKDE.2020.2981314.

[95] V. Yadav and S. Bethard, "A Survey on Recent Advances in Named Entity Recognition from Deep Learning models," in *Proceedings of the 27th International Conference on Computational Linguistics*, Santa Fe, New Mexico, USA, Aug. 2018, pp. 2145–2158. [Online]. Available: https://aclanthology.org/C18-1182

[96] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF Models for Sequence Tagging," *ArXiv*, vol. abs/1508.01991, 2015.

[97] Y. Gao, R. Wang, and E. Zhou, "Stock Prediction Based on Optimized LSTM and GRU Models," *Scientific Programming*, vol. 2021, pp. 1–8, Sep. 2021, doi: 10.1155/2021/4055281.

[98] N. Banik and Md. H. H. Rahman, "GRU based Named Entity Recognition System for Bangla Online Newspapers," in *2018 International Conference on Innovation in Engineering and Technology (ICIET)*, Dhaka, Bangladesh, Dec. 2018, pp. 1–6. doi: 10.1109/CIET.2018.8660795.

[99] S. Yan, J. Chai, and L. Wu, "Bidirectional GRU with Multi-Head Attention for Chinese NER," in *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, 2020, pp. 1160–1164. doi: 10.1109/ITOEC49072.2020.9141551.

[100] Z. Yang, R. Salakhutdinov, and W. W. Cohen, "Multi-Task Cross-Lingual Sequence Tagging from Scratch," *CoRR*, vol. abs/1603.06270, 2016, [Online]. Available: http://arxiv.org/abs/1603.06270

[101] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "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*, San Diego, California, Jun. 2016, pp. 260–270. doi: 10.18653/v1/N16-1030.

[102] X. Ma and E. Hovy, "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, Aug. 2016, pp. 1064–1074. doi: 10.18653/v1/P16-1101.

[103] R. Panchendrarajan and A. Amaresan, "Bidirectional LSTM-CRF for Named Entity Recognition," in *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*, Hong Kong, 2018. [Online]. Available: https://aclanthology.org/Y18-1061

[104] A. Goyal, V. Gupta, and M. Kumar, "Recurrent Neural Network-Based Model for Named Entity Recognition with Improved Word Embeddings," *IETE*

Journal of Research, pp. 1–7, Dec. 2021, doi: 10.1080/03772063.2021.2006805.
[105] H. Soltau, I. Shafran, M. Wang, and L. E. Shafey, "RNN Transducers for Nested Named Entity Recognition with constraints on alignment for long sequences," 2022, doi: 10.48550/ARXIV.2203.03543.
[106] D. S. Soper, "Greed Is Good: Rapid Hyperparameter Optimization and Model Selection Using Greedy k-Fold Cross Validation," *Electronics*, vol. 10, no. 16, p. 1973, Aug. 2021, doi: 10.3390/electronics10161973.
[107] A. A. Chowdhury, M. A. Hossen, M. A. Azam, and M. H. Rahman, "DeepQGHO: Quantized Greedy Hyperparameter Optimization in Deep Neural Networks for on-the-Fly Learning," *IEEE Access*, vol. 10, pp. 6407–6416, 2022, doi: 10.1109/ACCESS.2022.3141781.
[108] H. Alizadeh, M. Yousefnezhad, and B. M. Bidgoli, "Wisdom of Crowds cluster ensemble," *IDA*, vol. 19, no. 3, pp. 485–503, Jun. 2015, doi: 10.3233/IDA-150728.
[109] N. C. Oza and S. Russell, "Online Ensemble Learning," PhD Thesis, University of California, Berkeley, 2001.
[110] J. Pennington, R. Socher, and C. D. Maning, "GloVe: Global Vectors for Word Representation," *GloVe: Global Vectors for Word Representation*. https://nlp.stanford.edu/projects/glove/ (accessed May 03, 2022).
[111] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, Jul. 2009, doi: 10.1016/j.ipm.2009.03.002.
[112] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers," *J. Mach. Learn. Res.*, vol. 1, pp. 113–141, Sep. 2001, doi: 10.1162/15324430152733133.
[113] C. Sutton and A. McCallum, "An Introduction to Conditional Random Fields," 2010, doi: 10.48550/ARXIV.1011.4088.
[114] N. Patil, A. Patil, and B. V. Pawar, "Named Entity Recognition using Conditional Random Fields," *Procedia Computer Science*, vol. 167, pp. 1181–1188, 2020, doi: 10.1016/j.procs.2020.03.431.
[115] K. S. Komariah and B.-K. Shin, "Medical Entity Recognition in Twitter using Conditional Random Fields," in *2021 International Conference on Electronics, Information, and Communication (ICEIC)*, Jeju, Korea (South), Jan. 2021, pp. 1–4. doi: 10.1109/ICEIC51217.2021.9369799.
[116] J. P. C. Chiu and E. Nichols, "Named Entity Recognition with Bidirectional LSTM-CNNs." arXiv, Jul. 19, 2016. Accessed: Oct. 03, 2022. [Online]. Available: http://arxiv.org/abs/1511.08308

[117] C. Lee, “LSTM-CRF Models for Named Entity Recognition,” *IEICE Trans. Inf. & Syst.*, vol. E100.D, no. 4, pp. 882–887, 2017, doi: 10.1587/transinf.2016EDP7179.
[118] G. Cenikj, B. Korousic Seljak, and T. Eftimov, “FoodChem: A food-chemical relation extraction model,” in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, Orlando, FL, USA, Dec. 2021, pp. 1–8. doi: 10.1109/SSCI50451.2021.9660161.
[119] A. Goyal, V. Gupta, and M. Kumar, “Recent Named Entity Recognition and Classification techniques: A systematic review,” *Computer Science Review*, vol. 29, pp. 21–43, Aug. 2018, doi: 10.1016/j.cosrev.2018.06.001.
[120] G. Fumera and F. Roli, “A theoretical and experimental analysis of linear combiners for multiple classifier systems,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 942–956, Jun. 2005, doi: 10.1109/TPAMI.2005.109.

Acknowledgement

Alhamdulillah, first and foremost, all praises are to Allah, the Merciful, the All Beneficent, for His showers of blessings throughout my life. And for this opportunity to study and research in the AI Lab with the best and kindest professor, Prof. Sin Bong-Kee. I would like to express my gratitude to my advisor, Professor Sin Bong-Kee, for his guidance and support throughout my Ph.D. studies in AI Lab. During my time in the Lab, I received a great amount of knowledge not only in the AI area, but also in other areas, which he shared with me for my future developments. Prof. Sin Bong-Kee is not only my mentors, but also wonderful parents, friends, and teachers. He is really sincere to all of his students, and I am grateful to have him as my advisor and life mentor.

Extending my thanks and deepest appreciation to my thesis defense committee: Prof. Ha-Joo Song, Prof. Chang-Soo Kim, Prof. Hoon-Hee Kim, and Prof. Kyeongbo Kong, for their constructive comments and valuable suggestions, which allowed me to revise my thesis thoroughly and appropriately.

Furthermore, I would like to thanks all my colleagues in Artificial Intelligence Lab: Yang Woo-Hee, Nasim, Nepo, Nodira and Lee Tae-Ho. My sincere gratitude also goes to my best friends: Ahmad Wisnu M, Sandi Rahmadika and Vanda PW (and wife, Ifrina) for their support and valuable advice. I had a lot of unforgettable and happy moment with them and thanks for always helping each other in all situation and condition.

Lastly, my very special thanks to my family, especially my wonder women in my life which is my mother Anik, my late father Robasa (*almarhum*), my sisters Siti Nurfaedah and Annita Siti Fatimah as well as my step father Bapak Ara for their continuous support and encouragement. I am fully aware that without their unwavering support and prayers throughout my life, I would not have gotten to where I am today in pursuit of my dream.

List of Publications (SCI Journals)

(1) Ariana Tulus Purnomo, **Kokoy Siti Komariah**, Ding-Bing Lin, Willy Fitria Hendria, Bong-Kee Sin, and Nur Ahmadi, "Non-contact Supervision of COVID-19 Breathing Behaviour with FMCW Radar and Stacked Ensemble Learning Model in Real-Time", IEEE Transactions on Biomedical Circuits and Systems, (SCI, ISSN: 1932-4545), Published on 19 July 2022.

(2) **Kokoy Siti Komariah** and Bong-Kee Sin, "Enhancing Food Ingredient Named-Entity Recognition with Recurrent Network-based Ensemble (ERN) Model", MDPI Journal of Applied Science, (SCIE, ISSN: 2076-3417), Published on 13 October 2022.

(3) Yudi Widhiyasana, Maisevli Harika, Fahmi Faturahman Nul Hakim, Fitri Diani, **Kokoy Siti Komariah**, and Diena Rauda Ramdania, "Genetic Algorithm for Artificial Neural Networks in Real-Time Strategy Games", International Journal on Informatics Visualization (JOIV), Vol. 6, No. 2, (SCOPUS, ISSN: 2227-9707). Published on 30 June 2022.

(4) **Kokoy Siti Komariah**, Ariana Tulus Purnomo, and Bong-Kee Sin, "SMPT: A Semi-Supervised Multi-Model Prediction Technique for Food Ingredient NER (FINER) dataset construction", MDPI Journal of Informatics, (ESCI, ISSN: 2227-9707). *Will published soon.*

(5) Ariana Tulus Purnomo, **Kokoy Siti Komariah**, Ding-Bing Lin, Willy Fitria Hendria, Bong-Kee Sin, and Nur Ahmadi, "Breathing Behavior Detection using Boosting Algorithm for Imbalance Dataset Collected with FMCW Radar", Hindawi Journal of Electrical and Computer Engineering, (SCIE, ISSN: 2090-0147). *Will published soon.*

List of Publications (Conference Papers)

(1) **Kokoy Siti Komariah** and Bong-Kee Sin, “BERT Pre-trained Models for Data Augmentation in Twitter Medical Named-Entity Recognition”, in Korea Computer Congress (KCC) 2021, Jeju, Rep. of Korea.

(2) **Kokoy Siti Komariah** and Bong-Kee Sin, “Medical Entity Recognition in Twitter using Conditional Random Fields”, in International Conference on Electronics, Information, and Communication (ICEIC) 2021, Jeju, Rep. of Korea.

(3) **Kokoy Siti Komariah** and Bong-Kee Sin, “Nutrition-based Food Recommendation System for Prediabetic Person”, in Korean Software Congress (KCC) 2020, Pyongchang, Rep. of Korea.

(4) **Kokoy Siti Komariah**, Woo-Hee Yang, and Bong-Kee Sin, “Achored-LDA: Topic Modeling with Word Representation for Medical Discourse on Twitter”, in Ubiquitous Computing and Web Information Technology (UCWIT) 2019, Andong, Rep. of Korea.

(5) Sandi Rahmadika, **Kokoy Siti Komariah**, Kyeongmo Lee, and Kyung-Hyune Rhee, “Collaborative Federated Learning with Blockchain”, in the 4th International Symposium on Mobile Internet Security (MobiSec 2019), Taichung, Taiwan.

(6) **Kokoy Siti Komariah** and Bong-Kee Sin, “Health State Modeling and Prediction based on Hidden Markov Models”, in 2019 11th International Conference on Ubiquitous and Future Networks (ICUFN), Zagreb, Kroasia.