國 立 清 華 大 學

碩 士 論 文

題目 半督導式中文特定類型具名實體擷取之研究

A Semi-Supervised Method for Extracting Instances of a Certain Type in Chinese

系別 資訊系統與應用研究所 組別 乙組

學號姓名 9565702 陳瑩綺 （Ying-Chi Chen）

指導教授 張俊盛 博士 （Dr. Jason S. Chang）

張智星 博士 （Dr. Jyh-Shing Roger Jang）

中華民國九十八年五月

致謝

這篇論文的完成，真是經過一波三折。首先要感謝張俊盛老師對我的細心指導，讓我了解如何設計研究方法，且不厭其煩地一句句幫我改英文。感謝仲淇學長常主動跑來提醒我一些研究的細節，讓我不至於茫然無措，也很謝謝鑑城學長在我撰寫論文時，撥空幫我細看文章的邏輯性，也時常教導我做研究的技巧。謝謝 NLP lab 的各位同學，在我研究有困難時主動提供意見，和我並肩作戰。

另外，很感謝張智星老師，在我一進實驗室就給我很多學習的機會，在我低潮時安慰鼓勵我。謝謝張智星老師、李俊仁博士、和 MIR lab 的博班學長，在我練習報告時，很有耐性地指出我投影片的錯誤。

兩年的碩士生涯，很高興可以交到許多好朋友，一同出遊、party、打電動、聊天。特別謝謝小鈺和天才茵，在我難過的時候，願意陪伴我，時常和我吃飯，聊心事。MIR lab 是最歡樂的實驗室，真得很慶幸可以在這裡學習。

謝謝我的男友，雖然在當兵，還是每天打電話來，一有假就衝來陪我，從不因為我要忙論文，就心生抱怨。感謝我的可愛貓咪「比比」，雖然你不會講人話，但始終在我身邊，就算我因為太忙，常把你自己關在房間裡，有時又忘了餵你吃東西，或忘了幫你清廁所，你都不會生我氣，我一回家，你總是會撒驕地跑來歡迎，讓我不覺得回家是件寂寞的事。

最後，謝謝我的家人，不因為我拖延畢業時間，就對我施加壓力。謝謝我的媽咪，還很擔心地帶我四處求神問卜，爸比也常打電話來安慰我說不需要趕著畢業。謝謝我妹在我回台南時，都陪我去逛街散心。

謝謝大家，沒有你們，這篇論文無法完成。

摘要

本論文描述半督導式資料抽取方法，自動抽取中文文字資料中，某領域下特定類型的具名實體。此方法能自動建立及標記語料庫，以此訓練機器學習模型，用以消除過去督導式機器學習方法中人工標記的限制。在訓練階段，我們利用一般用途且易取得的同義詞典，從中選取一組種子於網路上取得語料，並利用種子自動標記取得的語料庫，再訓練機器學習模型於自動標記的語料庫。在執行階段，應用訓練好的機器學習模型，從自然語言書寫的文章，抽取出目標的具名實體。我們利用完全比對的評估方式，證明本方法可以有效地抽出目標的具名實體，最高正確率達 78%。此實驗結果以少量的種子資料，成功地消除人工標記的限制，顯示本方法的領域移植性優於其它督導式機器學習方法。

關鍵詞：資料擷取，具名實體辨識，網路語料庫，Maximum Entropy model (ME)，自動標記

ABSTRACT

We introduce a semi-supervised method for the extraction of instances of a certain type from a Chinese text under a domain. In our approach, a machine learning model for extraction is trained on an automatically collected and tagged corpus, aiming at eliminating the limiting factor of human annotation on current supervised systems. The method involves selecting seed data of target instances from off-the-shelf general purpose thesauri, using seeds to automatically collect a corpus from the Web, automatically tagging the corpus by seed data and training a machine learning model on the corpus. At run time, a natural language text is segmented into words, and the trained model is applied on the words to make the best tagging decisions, from which we extract target instances. The evaluation of exact match on a set of annotated test data shows that the method successfully extracts target instances at the precision rate of 78%. Our methodology accomplishes the elimination of human annotation on training data by small amount of seed data, and the method is highly portable to other domains.

Keywords: Information extraction (IE), Name Entity Recognition (NER), Web corpus, Maximum Entropy model (ME), automatically tagging

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Information extraction (IE) aims at automatically extracting structured information from unstructured machine-readable documents (e.g. a restaurant, or an electronic product). Every day, many natural language documents (e.g. blog articles) are created to the World Wide Web (WWW), and the fast growing amount of information on the Web make IE ever more important because the techniques of IE provide a shallow form of text understanding that extracts substrings about pre-specified types of entities or relationships from documents and web pages, in order to cope with such huge amount of data efficiently. Extracting instances of a specific type is one of the tasks of IE since the set of instances from the same category can be arranged in a structured format. An important part of instances of a certain type is named entities, which are the real things or instances in the world that are themselves natural and notable class members of subject concepts. For example, in the furniture, there are members like “地中海風格餐桌” or “和室桌” that are named entities.

Many systems that deal with the extraction of named entities such as Libra (libra.msra.cn) focus on very restricted pre-defined entity classes. Names of “persons”, “locations”, “organizations” are the three most popular types in the field of named entity recognition mainly for the reason that there are plenty available annotated corpora prepared by some competitional conferences, which we will discuss in the next section. The systems built upon the large annotated training data can achieve very high accuracy on the pre-defined classes. However, in the real world, people are interested in a variety of named entities that fall outside these pre-defined classes, especially for application of accessing information on WWW. Current pre-defined

classes cannot be applied to a wide range of domains in the WWW and it is a time-consuming task to annotate a large corpus for training. These application systems could be built up more efficiently if there is a way of using unannotated data for development.

Consider the situation of developing a NER system for a new domain, restaurants where it is important to identify dishes (e.g. “招牌牛肉麵特餐”). The best training process for the domain are probably not through the preparation of annotated data, which are very costly and time-consuming, but rather by crawling and automatic tagging a domain-specific corpus collected from the Web. A good way to collect and tag a corpus could be using a set of seed data, such as {“墨西哥玉米煎餅”, “香橙土司”, “西班牙炒飯”, “牛肉燉湯”, “五香燻牛肉”, “麵餅”, “豆醬”}. Seeds can be sent to a search engine (e.g. Google) to collect documents and recombined into regular expressions to identify the positions of domain-specific instances in the corpus based on the nature of named entities as combination of words. If seed data is segmented into words as {“墨西哥” “玉米” “煎餅”, “西班牙” “炒” “飯”, “牛肉” “燉” “湯”, “五香” “燻” “牛肉” , “麵餅”, “豆醬”}, the instance “牛肉玉米炒飯” can be matched by applying the regular expression /(墨西哥|西班牙|牛肉|五香|麵餅|豆醬) (玉米|煎餅|炒|飯|燉|湯|燻|牛肉|麵餅|豆醬)$^{+}$ | (麵餅 | 豆醬)$^{+}$)/. Intuitively, by applying the regular expression of seed data, we can identify most of instances in the given type because the components (words) in each instance are actually exchangeable to form another name. With an automatically tagged corpus, we can apply machine learning for the extraction of instances of a certain type.

We present a semi-supervised method that can automatically collect and tag a corpus from the Web expected to develop a named entity recognition (NER) system based on machine learning, such as hidden Markov model (HMM), maximum entropy hidden Markov model (MEMM), support vector machine (SVM), conditional random

Figure 1. An example applying our approach to extract foods from an article

field (CRF). In this paper, we choose to use MEMM as the machine learning model. An example session where the proposed method extracts instances of food from an article based on the MEMM model trained from automatic tagged corpus is show in Figure 1. The results are produced by an MEMM model trained on automatically constructed tagged corpus. We apply a heuristic method to tag each token as B-x (begin), I-x (inside), E-x (end), or O (outside), where x indicates the category. More specifically, during the learning process, the corpus is obtained from the Web and segmented to produce a sequence of tokens. Then, the regular expression of seed data (e.g. /(墨西哥|西班牙|牛肉|五香|麵餅|豆醬) (玉米|煎餅|炒|飯|燉|湯|燻|牛肉|麵餅|豆醬)$^{+}$ | (麵餅 | 豆醬)$^{+}$)/) matches the strings of domain-specific instances. We assign

each token one of BIEO tags based on its position in the matched strings. For example, the passage “要給我吃牛肉玉米炒飯”, segmented as “要 | 給 | 我 | 吃 | 牛肉 | 玉米 | 炒飯”, and tagged as {O O O B-food I-food E-food}. Based on the tagged corpus, a machine learning model is trained to automatically learn the features of target instances. We describe the training process in more detail in Chapter 3.

At run-time, our method starts with a natural language article submitted by the user, as the input article in Figure 1. The article is then segmented into words using a Chinese segmentation parser and each word is assigned one of BIEO tags based on the machine learning model trained specifically for the type that an user wants to extract. In our prototype, our method presents the extracted instances of a given type for the user (see Figure 1); alternatively, the instances extracted by our method can be used to build the index of a vertical search engine, the answers of a question answering system, or other systems that require the information in a certain category.

The rest of the thesis is organized as follows. We review the related work in the next chapter. Chapter 3 presents the proposed method for automatic tagging strategy and training process. Chapter 4 describes experimental setting and discusses the results of the experiment. Chapter 5 concludes and points out future research directions.

Chapter 2 Related Work

Information extraction (IE) has been an area of active research. Recently, information explosion intensifies the need for developing IE systems that help people to cope with the enormous amount of data available on the Web because the goal of IE is to produce structured data from natural language text, which refers to categorize contextually and semantically well-defined data in a certain domain (Manning et al., 2009). In our work, we address an aspect of information extraction that focus on extracting instances of a certain type in a domain. We not only identify general terms (e.g., “table” for furniture) but also named entities (e.g., “Taipei beef noodles” for food). General terms can often be looked up in a dictionary. On the other hand, named entities are newly created and hence not included in dictionaries; however, named entities usually carry the major information in a text (e.g., consider the situation of “dishes of a restaurant”)

More specifically, we focus on the named entity recognition (NER), also known as entity extraction or entity identification, namely, seeking to locate and classify names, which often show up as multi-words units, into a certain category. NER has long been an active topic of information extraction. In the expression “Named Entity”, the word “Named”, defined by S. Kripke (1982), aims to restrict the task to only those entities for which one or many rigid designators stands for the referent. Rigid designators include proper names as well as certain natural kind terms like biological species and substances. For instance, the automotive company created by Henry Ford in 1903 is referred to as Ford or Ford Motor Company (Nadeau & Sekine, 2007). One of the first research papers in the field presented by Rau (1991) describes a system to extract and recognize company names by heuristics and handcrafted rules. In general,

early works on NER mainly take the rule-based approach and rely on manually constructed patterns to achieve high precision, but the approach lacks robustness and portability. In contrast, we take the machine learning approach. Machine learning is more attractive because it is more portable and less expensive to maintain.

Since 1996 when Message Understanding Conference 6 (MUC-6) introduced the task of NER on the seven types: names of “persons”, “locations”, “organizations”, “dates”, “times”, and “percentage”, the amount of NER researches accelerate, especially for the recognition of “persons”, “locations”, “organizations”. A number of conferences based on common tasks such as MUC-6 has seen organized to encourage NER research, including IREC (Sekine & Isahara, 2000), CONLL (Tjong et al., 2003). Based on large-scale annotated corpora, machine learning approaches are widely adopted. Borthwick et al. (1998) describe a system that integrates the results of systems based on handcrafted rules or statistic methods into on the basis of the maximum entropy (ME) model and obtained a high accuracy. Uchimoto et al. (2000) introduce a NER method based on the ME model, and report that the optimal number of preceding/subsequent contexts to be incorporated in the model is two morphemes to both left and right from the current position. Bender et al. (2003) present a set of features which are easily obtainable for almost any language in ME models, such as lexicons, dictionary features. Similarly, our work also choose ME as our machine learning model, for the reason that as above authors remark, ME is a flexible statistic model that enable researchers to concentrate on finding features that characterize the problem. In contrast, although we adopt some of the features in above works, we also discover other effective features, which will be described in detail in Chapter 3.

Chinese named entity recognition is more difficult because the Chinese text lack word boundary, morphological variety, and capital letters to denote proper names and sentence breaks. Additionally, the flexibility of Chinese syntax structures makes

Chinese NER more complex than the English case. Recently, researchers on Chinese NER tend to use hybrid models to cope with the complexity in Chinese text; that is, they either incorporate rules into machine learning models or perform post-processing to improve the accuracy. Tsai et al. (2004) present a hybrid model that incorporate rule-based knowledge and templates into a ME framework. Zhang and Zhang (2007) present a hybrid model for automatic Chinese person name and location name recognition based on the ME model and post-processing. These previous works of supervised machine learning achieve high performance of NER are all based on the manually prepared large annotated training data and are limited to pre-defined classes (e.g., persons, organizations, locations). However, very few NER systems have been developed to handle online text with the requirements of low cost, flexibility, and ease of adaptation to new domains, in order to deal with huge amount of data on the WEB. The preparation of corpora is expensive and time-consuming. In contrast to previous work, we aim at developing a strategy of automatically collecting and tagging a corpus to cope with the problem.

Recently, some researchers have recognized the problem of restricted domains on supervised machine learning system. Sekine et al. (2002) extend named entity hierarchy to include about 150 NE types which may cover most of the entities which appear in usual newspaper articles. Zhao and Liu (2008) present a system that recognize Chinese product NER such as "*摩托羅拉 V8088 折疊手機*" (*Motorola V8088 clamshell cell phone*) by applying hierarchical hidden Markov model on an annotated corpus. However, it is time-consuming for them to prepare an annotated corpus because of the lack of training data for product NER.

In the NER study closer to our work, the surface pattern approach involves semi-supervised learning using a small set of seeds and bootstrapping process. Hearst (1992) pioneered the surface pattern approach using three handcrafted rules and

bootstrapping for the automatic acquisition of the hyponym lexical (is-a) relation from unrestricted texts. Riloff and Jones (1999) introduce a mutual bootstrapping technique that uses a handful of seed words for a category and unannotated training texts. Their goal is to learn two dictionaries for NER: a dictionary of semantic lexicons and a dictionary of extraction patterns for the domain. Similar to our work, the surface patterns approaches do not need annotated corpora. Instead, a small set of seeds are used to find more NE instances in a large-scale corpus.

Recent work on the surface pattern approach has been applied on the Web corpus. Brin (1998) uses lexical features represented in regular expressions in order to generate lists of book titles paired with book authors from the World Wide Web (WWW). Patwarhan and Riloff (2006) collect a domain-specific corpus from the Web by handcrafted queries and bootstrap to learn domain-specific surface patterns. Similarly, our method also collects a domain-specific corpus from the Web for the purpose of collecting a corpus, but we do not use handcrafted queries. Instead, we take the seeds from general purpose thesaurus (e.g. Wordnet) and directly use them as queries, to avoid too much human intervention. Previous work on surface patterns perform well based on highly accurate rules produced mainly by outer context information, meaning fixed patterns made by strings, and thus lack of flexibility. They achieve high precision at the expense of low recall. Therefore, surface patterns are most successful to discover named entities instead of identifying entities. Beside purely surface patterns, Downey et al. (2007) locate complex named entities in Web text by computing Pointwise Mutual Information and Information Retrieval (PMI-IR), developed by Turney (2001), based on the features like Web page counts and capital words. Downey's approach can work on the named entities beside the pre-defined classes (e.g. names of person, organization, location); however, it is used to create large lists of named entities, not designed for resolving ambiguity in a given

document; additionally, the useful feature of capitalization in English in Downey's work cannot apply in Chinese case.

In a study more closely related to our work, Nadeau et al. (2006) propose a NER system that combines named entity extraction based on HTML surface patterns with a simple form of named entity; that is, they first start with a set of seeds to learn HTML patterns for the extraction of more entities, and then a list formed by the extracted entities is heuristically used to disambiguate entities in a document, such the list lookup strategy, and multi words units with capital words. In contrast, our method obtains training data from the Web and use machine learning to train a NER system, instead of using heuristic rules. Shinzato et al. (2006) describe an automatic dictionary construction method for NER for a specific domain such as restaurant guides. Their dictionary construction method exploits the co-occurrence strength of two expressions in HTML itemizations calculated from average mutual information. Then, the dictionary entries are used as features to train a NER system based on Support Vector Machine (SVM). Similarly, we also apply machine learning strategy to build a NER system. However, their method still needs a small annotated corpus, while our approach applies automatic tagging.

In contrast to the previous research in NER, we present a method that can identify named entity, instances, of a certain type with the goal of reducing the degree of human intervention and enhancing the portability from one domain to another. We exploit the semi-supervised procedure by collecting a corpus from the Web with the seed data from general purpose thesaurus, and automatically tagging the corpus to produce training data for machine learning.

Chapter 3 Method

Most named entity recognition (NER) research focus on several predefined classes (e.g., persons, organizations, locations) and rely on annotated corpora. These supervised machine learning can achieve high accuracy, but at the same time, the human labor is the limiting factor. Unfortunately, predefined classes and annotated corpora can not cope with the wide range of domain-specific instances on the Web. In the real world, these named entities falling outside of the predefined classes are most interesting and carry much information in a domain (e.g. dishes of a restaurant, products of a furniture store). Such useful information may be costly to extract based on supervised machine learning methods. To reduce the effort of human annotation, which is the major limiting factor of machine learning, we propose a promising approach for automatically collecting and tagging training data in a format that allow the machine learning approaches to train a model expected to identify instances of a certain type in a domain.

3.1 Problem Statement

We focus on the extraction of instances of a certain type in a domain: extracting from a natural language text a set of instances of a certain type in a domain. The extracted instances can be examined by a human user directly, or passed on to a vertical search engine as index. Thus, it is crucial that the instances of a certain type, including general terms and named entities, can be extracted. At the same time, the quality of extracted instances should meet a certain degree of accuracy. Therefore, our goal is to extract a set of instances in a certain type of domain that at the same time should not contain too many false positive results. We now formally state the problem that

(1) Collect a domain-specific corpus from the Web (Section 3.2.1)
(2) Tag automatically a corpus to constitute training data (Section 3.2.2)
(3) Select the features for machine learning (Section 3.2.3)
(4) Train a machine learning model

Figure 2: Outline of the training processes

we are addressing.

Problem Statement: We are given a natural langue text *Txt* that contains instances of a certain type in a domain and a set of seed data *S*. Our goal is to extract a set of instances *Ins* of the type from *Txt*. For this, we use *S* and information from general purpose thesaurus (e.g. Wordnet) to collect a corpus *C* and use heuristics to automatically tag *C* such that a machine learning approach can be applied to train a model for extracting *Ins*.

In the rest of this section, we describe our solution to this problem. First, we define a strategy for automatically collecting and tagging a domain-specific corpus (Section 3.2). This strategy relies on a set of seed data, derived from general purpose thesauri (which we will describe in detail in Section 4.1). In this section, we also describe how to annotate raw training data and how a machine learning approach is used to train a model. Finally, we show how our method extracts target instances at run-time based on the machine learning model (Section 3.3).

3.2 Training a machine learning model for extraction

We attempt to collect a domain-specific corpus from the Web and automatically tag it with heuristics to constitute data to train a machine learning model for the extraction of instances of a certain type in a domain. Our learning process is shown in Figure 2.

3.2.1 Data Collection

In the first stage of the learning process (Step (1) in Figure 2), we collect from the Web a domain-specific corpus in which most of sentences contain target instances. For example, to collect a corpus for information extraction related to foods, the sentences should look like "你能想像出長達2400米的墨西哥玉米煎餅嗎？" which contains the target instance "墨西哥玉米煎餅". This kind of sentences can be retrieved by using a set of seeds to query a search engine (e.g., *Google*). By using a seed "墨西哥玉米煎餅", we are likely to retrieve such the sentences in returned snippets, and sentences with other target instances may also be retrieved in the same snippet based on the empirical experience that similar instances usually show up nearby.

The input of this stage is a set of seeds of a certain type. These seeds are sent as queries to a search engine to retrieve texts to build a corpus for training purpose. As we will describe in Section 4.1, general purpose thesauri (e.g., Wordnet) contain synonymy and antonyms, defining different kinds of terms and relationships; thus, instances of a certain type in the domains of interest can be found in a thesaurus. For instances, we can select instances foods from Wordnet as seed data. For seed data, we use a number of seeds *numSeed*s with length longer than *minQLength* characters as queries for a search engine because short instances are more likely to be a part of another instance in a sentence, which may introduce noise into the data and create problem for the subsequent stage of automatic tagging. For example, if a short instance "玉米" retrieves a sentence "你能想像出長達2400米的墨西哥玉米煎餅嗎？", in which "玉米" is as a part of the long instance "墨西哥玉米煎餅", the program may wrongly treat the words except "玉米" as outside of a target instance.

The output of this stage is a set of snippets that contain queried seeds returned by a search engine (e.g., *Google*). Some sample queried seeds and retrieved snippets are

seed	snippet
墨西哥玉米煎餅	你能想像出長達 2400 米的**墨西哥玉米煎餅**嗎？美國內華達州里諾市近日展出了這麼... 據報導，這個巨型**墨西哥玉米煎餅**總共耗費了 8200 多個**玉米麵餅**、2000 磅的**香炒豆醬**以及...
	小心翼翼地打開錫紙包裝，警方人員驚訝地張大了嘴巴，這件所謂的危險武器，竟然是一個　巨大的**墨西哥玉米煎餅**。這個**煎餅**中間裹著大量的**牛排**、**生菜**和**墨西哥胡椒粉**。

Table 1: Examples of retrieved snippets by a seed

shown in Table 1, in which the bold characters present the target instances foods. Note that in the examples of Table 1, the retrieved sentences not only contain seeds but also other foods, such as “玉米麵餅” and “香炒豆醬”. We will describe in the next section the automatic tagging procedure of the sentences in the returned snippets to produce training data.

Although there are other ways to prepare a corpus, our method uses small amount of seed data obtained from off-the-shelf knowledge sources to expand training data quickly. This procedure costs relatively less human effort and are language independent.

3.2.2 Tag automatically a corpus

In the second stage of training, we automatically tag the corpus collected from previous stage, to produce training data for machine learning (Step (2) in Figure 2). In other words, given a corpus, we assign to each token (word) one of B, I, E and O tags. In this scheme, the beginning token of an instance is tagged as B. Subsequent tokens within the instance are tagged as I. The ending token of an instance is tagged as E. All other tokens are tagged O. The B, I and E tags are suffixed with the type of the instances, such as B-type, I-type, E-type. For example, if the sentence “你能想像出長達 2400 米的墨西哥玉米煎餅嗎？” is segmented as “你 | 能 | 想像出 | 長達 |

```
Procedure AutoTagCorpus(Corpus, Seed)
   TagResult = []
1) RE = GenerateRegularExpression(Seed)
   For each snippet_i in Corpus
2)    SegmentedSnippet = ApplyChineseSegmenter(snippet_i)
3)    MatchedResult = ApplyREFindAll(SegmentedSnippet, RE)
4)    TagResult[i]= a list of "O" with the same size of SegmentedSnippet
      For each MatchedString in MatchedResult
        For each word_j in MatchedString
5a)        If word_j = the first word of MatchedString
             TagResult[i][j] = "B"
5b)        Else if word_j = the last word of MatchedString
             TagResult[i][j] = "E"
5c)        Else
             TagResult[i][j] = "I"
```

Figure 3: Automatically tagging the corpus

2400｜米｜的｜墨西哥｜玉米｜餅｜嗎｜？”, we assign a tagging sequence as {O, O, O, O, O, O, O, B-Food, I-Food, E-Food, O, O}. Assigning tags has become the standard way to identify the boundaries of specific types of word sequences as the example above. The procedure of our automatically tagging is shown in Figure 3.This procedure takes the corpus and seed data acquired from the last stage as input.

In Step (1) of the algorithm, we use seed data, selected from general purpose thesauri for collecting a corpus in the last stage, to generate a regular expression *RE* for the tagging function. Recall that we assume the components (words) of each instance are exchangeable to form another instance. For example, the components of the instances {“墨西哥玉米煎餅”, “香橙土司”, “西班牙炒飯”, “牛肉燉湯”, “五香燻牛肉”, “麵餅”, “豆醬”} can be used to form another name “牛肉玉米炒飯”. For this, all the seed data are segmented into words (tokens), where first words of each

seed are selected as beginning tokens *Btoken*, the subsequent tokens as inside tokens *Iotken* and the last tokens as ending tokens *Etoken*. When an instance is only composed by one word, its token is additionally selected as single tokens *Stoken*. As the example above that seed data segmented as {“墨西哥”“玉米”“煎餅”, “西班牙”“炒”“飯”, “牛肉”“燉”“湯”, “五香”“燻”“牛肉”, “麵餅”, “豆醬”}, then *Btoken*={“墨西哥”, “西班牙”, “牛肉”, “五香”}, *Itoken*={“玉米”, “炒”, “燉”, “燻”}, *Etoken*={“煎餅”, “飯”, “湯”, “牛肉”}, and *Stoken*={“麵餅”, “豆醬”}

After the segmentation and simple classification of tokens according to their position in an instance, *RE* is produced in the form as / ((*Btoken*|*Stoken*) (*Itoken*|*Etoken*|*Stoken*)$^{+}$) | (*Stoken*)$^{+}$ /. There are two parts in *RE*. The first part ((*Btoken*|*Stoken*) (*Itoken*|*Etoken*|*Stoken*)$^{+}$) contains two subparts. The first subpart (*Btoken*|*Stoken*) contains the prefixes on singles or the instances (e.g., “墨西哥”, “麵餅”). The second subpart contains *Itoken*, *Etoken* and *Stoken*, for the reason that in Chinese instances, *Itoken* and *Etoken* are usually interchangeable. However, *Btoken* are not part of this second subpart because most of the times, *Btoken* are as pre-modifiers instead of repetitive, interchangeable words for the type. For example, “墨西哥” does not mean a kind of foods when not connected to “玉米”. Furthermore, *Btoken* often represent the left boundary of an instance, so it is necessary to put *Btoken* in the first position. In addition, single tokens, since they are composed by only one word, is necessarily head token for the type, such as “麵餅”; therefore, in the principle of exchangeability, *Stoken* are put into the second portion with *Itoken* and *Etoken*. In Chinese syntax, two word of the same type can be connected together, so *Stoken* is also allowed to be in the first portion with *Btoken*, making this rule a little bit loose to match more instances. Note that the first part of *RE* only considers instances of two words or longer, namely one *Btoken* or *Stoken* plus at least one *Itoken* or *Etoken* or *Stoken* since the major components in this part are derived from

original snippet	你能想像出長達 2400 米的墨西哥玉米煎餅嗎？美國內華達州里諾市近日展出了這麼... 據報導，這個巨型墨西哥玉米煎餅總共耗費了 8200 多個玉米麵餅、2000 磅的香炒豆醬以及...
segmented result	你 能 想像出 長達 2400 米 的 墨西哥 玉米 煎餅 嗎 ？ 美國 內華達州 里 諾 市 近日 展出 了 這麼 ... 據 報導 ， 這個 巨型 墨西哥 玉米 煎餅 總共 耗費 了 8200 多 個 玉米 麵餅 、 2000 磅 的 香 炒 豆醬 以及 ...

Table 2: An example of segmented snippet

instances of multi-words. The second part of RE $(Stoken)^+$, which is much simpler, matches one or more *Stoken* because *Stoken* surely belong to the target type as the discussion above, and the Chinese syntax structures allow corresponding concepts freely combined to express combinational meaning. By using the example in the previous paragraph, *RE* looks like /(墨西哥|香|西班牙|牛肉|五香|麵餅|豆醬) (玉米|煎餅||炒|飯|燉|湯|燻|牛肉|麵餅|豆醬)$^+$|(麵餅|豆醬)$^+$/.

Once *RE* is produced using the seed data, for each snippet in the corpus, we begin our procedure of automatically tagging. First, we initialize *TagResult* to store the tagging result of the whole corpus. In Step (2) of Figure 3, a snippet $snippet_i$ is segmented, separating each word by a blank, as the example in Table 2. In Step (3), *RE*, generated in Step (1), is applied to *SegmentedSnippet* to obtain all the matched strings.

After applying *RE*, we begin to automatically tag the snippet. In Step (4), we initialize *TagResult*[i] to a list of "O" to store the tags of all the word in $snippet_i$. In Step (5a) to (5c), we examine each *MatchedString* in *MatchedResult*, in order to assign B, I, or E tags to each word $word_j$. Note that *j* corresponds to the position of current word in $snippet_i$; therefore, *TagResult*[i][j] stores the value of tag of current word $word_j$. In Step (5a), the first word of current *MatchedString* is tagged as "B", the

original snippet	你能想像出長達2400米的墨西哥玉米煎餅嗎？美國內華達州里諾市近日展出了這麼... 據報導，這個巨型墨西哥玉米煎餅總共耗費了8200多個玉米麵餅、2000磅的香炒豆醬以及...
tagging result	你 能 想像出 長達 2400 米 的 **墨西哥 玉米 煎餅** 嗎 ？ O O O O O O O **B-food I-food E-food** O O
	美國 內華達州 里 諾 市 近日 展出 了 這麼 ... 據 報導 ， O O O O O O O O O O O O O
	這個 巨型 **墨西哥 玉米 煎餅** 總共 耗費 了 8200 多 個 O O **B-food I-food E-food** O O O O O O
	玉米 麵餅 、 2000 磅 的 **香 炒 豆醬** 以及 ... **B-food E-food** O O O O **B-food I-food E-food** O O

Table 3: The tagging result of the sample snippet in Table 2

last word as “E” (Step (5b)), and the rest of words as “I” (Step (5c)). A sample tagging result of the snippet in Table 2 is shown in Table 3, in which words in bold represent *MatchedString*.

By using the simple procedure of regular expression described above, the result of this training stage is a set of training data, automatically tagged by assigning each token one of BIEO tags to produce valid data for machine learning.

3.2.3 Apply machine learning on tagged data

In the third and final stage of training, given a tagged corpus, we apply a machine learning strategy to learn features of each BIEO tags for the computation of conditional probability of a word with a tag. We choose Maximum Entropy model (ME) as our machine learning strategy.

ME is a flexible statistical model that offers a clean way to combine diverse pieces of linguistic contextual evidence in order to estimate the probability of an *outcome* occurring with a certain linguistic context *history*. In our method, outcome

space is comprised of BIEO tags for an ME formulation of extraction of instances. ME computes the probability $p(o|h)$ for any o from the space of all possible outcomes O, and for every h from the space of all possible histories H. A *history* is all the conditional data that enable one to assign probabilities to the space of outcomes. In our problem, *history* could be viewed as all information derivable from the test corpus relative to the current token. The computation of $p(o|h)$ in ME depends on a set of binary-value *features*, which are helpful in making a prediction about the outcome. For instance, one of our features is: when current token is a known ending token of food, it is likely to be tagged as "E-food". More formally, we can represent this feature as:

$$f(h,o) = \begin{cases} 1: \text{if End\_Char\_Of\_food}(h) = \text{true and } o = E-food \\ 0: \text{else} \end{cases} \quad (1)$$

In this formula, *End_Char_Of_Food*(*h*) is a binary function that returns the value *true* if the *current word* of the history *h* is in the list of known ending tokens. For example, if the list of ending tokens contains {"飯", "麵", "排", "湯"} and the current word is "湯", the value of this feature of "湯" with the tag "*E-food*" is 1, or in other words, active.

Given a set of features and a training corpus, the ME estimation procedure generates a model where every feature f_i has associated with its weighting parameter α_i. This allows us to compute the conditional probability as follows:

$$p(o|h) = \frac{1}{Z(h)} exp(\sum_{i=1}^{n} \alpha_i f_i(h,o)) \quad (2)$$

where $p(o|h)$ denotes the conditional probability of predicting an outcome o with a history h, o is the outcomes BIEO tags, h is all the conditional information derived from current word, $\frac{1}{Z(h)}$ is the normalization factor, f_i is a binary feature function, α_i is the weight of f_i. Intuitively, the probability is the multiplication of weights of active features, meaning those $f_i(h,o)=1$. The weight α_i is estimated by a procedure called

Generalized Iterative Scaling (GIS), which is an iterative method that improves the estimation of the weights at each iteration, in order to maximize the model's log-likelihood. The ME estimation technique guarantees that for every feature f_i, the expected value of α_i equals the empirical expectation of α_i in the training data.

As many researchers have remarked, the major advantage of ME is that it allows the modeler to concentrate on finding the features that characterize the problem while letting the ME estimation systematically deal with assigning relative weights to the features (Borthwick, 1998). In addition, ME has the ability to incorporate any binary-valued features from different knowledge sources and can handle more features than other approaches, such as hidden Markov Model (HMM), Conditional Random Field (CRF), because ME does not require enumeration of the space of all possible observations, by which ME computation has less limitation on computer memory (Wu et al., 2006). Therefore, we choose ME as the machine learning strategy in this thesis.

To build a ME model, we need to identify features that best characterize the problem of extracting instances of a certain type. In the following, we discuss each feature we used in turn:

1. Word-class features:

   Word-class feature is the most important feature we use. Based on the spirit of the regular expression of BIEO tags, we categorize words into three classes of BIE in advance and use list lookup strategy to select the word class as features of a word. Seed data acquired from the data collection stage is segmented into words. Each word according to its position in its original instance is categorized into the corresponding class. First words of each seed are put into a list *BtokenList*, last words into *EtokenList*, and the rest words into *ItokenList*. Single tokens, derived from instances composed only by one word which is thus sure to be in the target

	Example
seed data	墨西哥玉米煎餅，西班牙炒飯，牛肉燉湯，五香燻牛肉，麵餅，豆醬
segmented data	墨西哥 ｜ 玉米 ｜ 煎餅，香 ｜ 橙 ｜ 土司，西班牙 ｜ 炒 ｜ 飯，牛肉 ｜ 燉 ｜ 湯，五香 ｜ 燻 ｜ 牛肉，麵餅，豆醬

Table 4: Examples of seed data and its segmented result

List	**Example**
Btoken	墨西哥，香，西班牙，牛肉，五香，麵餅，豆醬
Itoken	玉米，橙，炒，燉，燻，麵餅，豆醬
Etoken	煎餅，飯，湯，牛肉，麵餅，豆醬
Bchar	墨，西，哥，香，西，班，牙，牛，肉，五，香，麵，餅，豆，醬
Ichar	玉，米，橙，炒，燉，燻，麵，餅，豆，醬
Echar	煎，餅，飯，湯，牛，肉，麵，餅，豆，醬
LastChar	餅，飯，湯，肉，餅，醬
Bigram	墨西哥玉米，玉米煎餅，西班牙炒，炒飯，牛肉燉，燉湯，五香燻，燻牛肉

Table 5: Sample lists of word-class, character-class, last characters and bigram features, derived from data in Table 4.

concept, are put into all three lists since the Chinese syntax structures allow corresponding concepts freely combined to express combinational meaning. Table 4 shows the example seed data and its segmented result. Table 5 shows the sample lists derived from seed data in Table 4. In Table 5, segmented seed data and lists of word-class are as below:

(1) Segmented seed data: {“墨西哥” “玉米” “煎餅”, “香” “橙” “土司”, “西班牙” “炒” “飯”, “牛肉” “燉” “湯”, “五香” “燻” “牛肉”, “麵餅”, “豆醬”}

(2) *BtokenList*: {“墨西哥”, “香”, “西班牙”, “牛肉”, “五香”, “麵餅”, “豆醬”}

(3) *ItokenList*: {“玉米”, “橙”, “炒”, “燉”, “燻”, “麵餅”, “豆醬”}

(4) *EtokenList*: {“煎餅”, “土司”, “飯”, “湯”, “牛肉”, “麵餅”, “豆醬”}

For example, “玉米” will have an active feature denoting its existence in *ItokenList*. All other words, such as “湯勺”, are considered not belonging to any

word class. Note that one token (word) could be in multiple classes, as “牛肉” in both *BtokenList* and *EtokenList*, which means it could be the beginning or ending word of an target instance. In this case, ME model will adjust its probabilities of “B-food” and “E-food” according to other features, such as features from its previous word, which we will describe later.

The benefit of word-class features is that it allows us to incorporate data from different knowledge sources. Intuitively, we directly use seed data from the first stage, but in practice, we can expand the word classes at will by adding other name lists, which will be discussed in Chapter 4.

2. Character-class features:

We extend the idea of word-class features by taking characters of each word in a class to form a character class. As in Table 5, *EtokenList* has {“煎餅”, “飯”, “湯”, “牛肉”, “麵餅”, “豆醬”}, and then we take its characters to make a *EcharList* as {“煎”, “餅”, “飯”, “湯”, “牛”, “肉”, “麵”, “餅”, “豆”, “醬”}. Assigning a class to each character in a token can help to cope with the problem of data sparseness. In the creation of a Chinese word in a certain type, some fundamental elements, as head words, are often used in combination of unseen characters (e.g, “旗” in “旗魚”). In the reality, it is also these head characters that let people realize which category a new word belong to. For example, “旗魚” is a kind of fish because of the character “魚”. In Chinese words, general characters that indicate certain categories tend to locate near the end of a word, as the previous example in which “魚” is the last character. In contrast, although the word “湯勺” has a general character “湯” which is often used in words of the category “food”, it is not a dish but a kind of household utensil. People can understand that “湯勺” does not indicate a dish mostly because “湯” does not

show up as the last character of the word, and the last character "勺" is a common element of utensils.

To characterize this nature, when classifying a character, we assign a number from the end of the word. For example, the *Char-1* of "煎餅" is "餅" and its *Char-2* is "煎". As the *EcharList* in Table 5, "煎餅" will have two active features, one for denoting *Char-1* in *EcharList* and the other for *Char-2* in *EcharList*. For another example of 湯勺", its *Char-1* will have an active feature denoting it is not in any character lists in Table 5. Note that we assign numbers from back to front, so the meaning of each number with regard to its position will not be confused by different length of words. In addition, to prevent using too rare elements, which are often such unique that does not indicate common characteristics of a type, we only select the characters over a threshold of counts *minCharCount* into lists.

3. Ending character features:

We take the last characters of each instance in seed data to form a list of ending characters. Note that in the character-class feature, we directly take characters of tokens in a class; therefore, *EcharList* not only contain ending characters but also other elements that show up in ending tokens, such as "煎" and "牛", which is not the ending characters, in *EcharList* of Table 5. In the examples of Table 5, we can form a list *LastCharList* {"餅", "飯", "湯", "肉" "餅", "醬"}, "濃湯" will have a feature indicating its existence in *LastCharList*, while "湯勺" will have a feature indicating its nonexistence in *LastCharList*. Last character features mainly identify the ending token of an instance. Intuitively, in Chinese, last characters are most characteristic of the semantic category. For example, typically dishes end with characters such as "湯" or "飯", while furniture names typically end with "桌" or "爐".

4. Bigram features:

Tokens from seed data are combined into bigrams to form a list *BigramList*, for the benefit that bigrams can present more specific information of a category than single tokens. Note that instances of single tokens can not form bigrams since they do not have combinational information inside their own instances. We check if bigrams of current token with its previous as well as next token are in *BigramList*. As in Table 5, if seed data are segmented as {“墨西哥” “玉米” “煎餅”, “香” “橙” “土司”, “西班牙” “炒” “飯”, “牛肉” “燉” “湯”, “五香” “燻” “牛肉” , “麵餅”, “豆醬”}, *BigramList*={“墨西哥玉米” “玉米煎餅”, “香橙” “橙土司”, “西班牙炒” “炒飯”, “牛肉燉” “燉湯”, “五香燻” “燻牛肉”}. For example, “煎餅” and its previous token is “玉米” has an active feature indicating the bigram is in *BigramList*, while “煎餅” and its next token as “嗎” has an active feature for the nonexistence of this bigram in *BigramList*.

Intuitively, the above four kind of features directly use the seed data to form lists. Furthermore, we can enrich these lists by adding other name lists, which can be easily gathered from the Web or dictionaries. We will examine this idea in the experiment.

5. Part-of-Speech features:

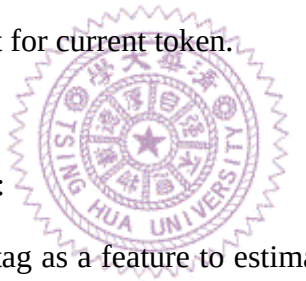
The part-of-speech of current token offered by a Chinese segmenter is directly used as a feature because part-of-speech presents syntactic information of a token. For example, the part-of-speech of current token “牛肉”, as a common noun (Na), is directly taken as a feature.

6. Semantic label features:

The semantic labels of Chinese thesaurus provided by CKIP of Academia Sinica are also taken directly as features because semantic labels offer basic concept categories of a word. For example, “meals”, as the semantic label of “牛肉” is taken directly as a feature.

7. Preceding and subsequent features:

As Uchimoto et al. (2000), we also take all the features from previous and following tokens within a certain window size *windowSize*, to provide surrounding information for the improvement of accuracy. For instance, at a window size of two, we take all the active features from two tokens on the left and two tokens on the right for current token.



8. Previous state features:

We take the previous tag as a feature to estimate transition probability in the model, for Viterbi search at run-time. This transition feature is for the procedure of Maximum Entropy Markov model (MEMM), which is the combination of ME and HMM to estimate state transition probability, in order to find better tagging sequence at run-time (McCallum et al., 2000). For example, if the previous tag of current token is “O”, then it has an active feature “*Current-PreTag-Is-O*”. At run-time, this feature will be dynamically changed according to which previous node we are computing (McCallum et al., 2000). The detail of MEMM procedure will be described in Section 3.3.

By selecting these features, we train a ME model for the extraction of target instances. The run-time procedure of which will be described in Section 3.3.

3.3 Run-time extraction of target instances

```
Procedure ExtractInstance(Article, MEmodel)
(1) segment = ChineseSegmenter(Article)
   queue=[]
   for each <word_i, pos_i> in segment
(2)    history_i=FindActiveFeature(word_i, pos_i)
     Append <word_i, history_i> to queue

   DPtable=[]
   BackTrackTable=[]
   For each <word_i, history_i> in queue
(3)    currentHistory=AddSurroundingHistory(history_i, WindownSize)
     HISTORY=[]
     For t = 1 to 4
(4)      HISTORY.append(currentHistory)
(4a)  Append "Current-PreTag-Is-B" to HISTORY[1]
(4b)  Append "Current-PreTag-Is-I" to HISTORY[2]
(4c)  Append "Current-PreTag-Is-E" to HISTORY[3]
(4d)  Append "Current-PreTag-Is-O" to HISTORY[4]

   DPtable[i]=[]
   BackTrackTable[i]=[]
   For each tag_j in BIEOtags
     tmpProbArr=[]
     For each <tag_t, preProb_t> in DPtable[i-1]
(5a)     MEMMprob= preProb_t*ComputeMEprob(HISTORY_t, tag_j, MEmodel)
(5b)     Append <MEMMprob, tag_t> to tmpProbArr
(5c)  <curProb,curPreTag>=GetMaxProbAndPreNode(tmpProbArr)
     Append <word_i, tag_j, curProb> to DPtable[i]
     Append <curPreTag> to BackTrackTable[i]

(6) TagSequence=ViterbiSearch(BackTrackTable)
(7) Results=ExtractInstance(TagSequence)
```

Figure 4: Extracting target instances at run-time

Once the machine learning model is trained for a certain type of instances, the model is used to find the best tagging sequence of an article, in order to extract the target instances. At run-time, we apply Maximum Entropy Markov model (MEMM) proposed by McCallum et al. (2000). Basically, MEMM is still a ME framework, but incorporates state-transition of HMM as one of features into the model, to substitute for the transition probability in HMM. It is shown that MEMM can find better tagging sequence than pure ME approaches (McCallum, 2000). Therefore, we choose MEMM at run-time to find the best tagging sequence. The procedure of run-time extraction is shown in Figure 4. The input of procedure is an article and a ME model trained for a certain domain, such as food.

In Step (1), an article is segmented into words by a Chinese segmenter as "緊接著｜是｜海膽｜花｜壽司". In Step (2), we extract the active features of each word, as the example that for the target instances food, the word "海膽" will have active features denoting its existence in *BtokenList*, its characters in *BcharList* and so on. For each word $word_i$, we store its active features into $hisotry_i$, which is a variable containing all the observations of $word_i$ for the estimation of ME probability.

We use *DPtable* a two-dimensional array for storing the conditional probabilities of each word with its four BIEO tags respectively, in order to implement Viterbi search. Let a sentence has *n* words, and then *DPtable* has a size of 4**n*. In *DPtable*, a node means a word with one of BIEO tags; therefore, each word has 4 nodes that store their MEMM probability. *BackTrackTable* is also a two-dimensional array, to store the best path of each node to its previous node for the principle of Viterbi search. The benefit of using Viterbi algorithm is that it let ME model decide all the conditional probabilities, and among these, it can find the globally optional tagging sequence of a sentence without trapping in local maximal solutions.

For each word $word_i$ and its $history_i$ in *queue*, we begin to compute their

probability for dynamic programming. In Step (3), preceding and subsequent features at a certain *WindowSize* are selected and stored with $history_i$ into *currentHistory*. In order to apply MEMM, in Step (4), before computing conditional probability by *currentHistory*, we need to make a one-dimensional array *HISTORY* in which each record represents the *history* of $word_i$ with a different previous state feature. Step (4a-d) add features of previous state as “B”, “I”, “E” or “O” respectively to each record of *HISTORY*. Making this array is in the spirit of dynamic programming of MEMM that each node needs to find its best path from previous node for state transition, so each node needs to compute the probabilities by four different observations in *HISTORY* respectively, and pick up the highest one as its probability.

After the preparation of *HISTORY*, we begin to compute conditional probability of each node; in other words, $word_i$ needs to compute four different probabilities of BIEO tags tag_j. *tmpProbArr* is to store the probabilities of four different previous state features of $word_i$ with tag_j. In MEMM, the probability of current node needs to be multiplied by that of previous node inferred by the previous state feature of $HISTORY_t$, so we go through each probability of previous word $preProb_t$, meaning the probabilities of $word_{i-1}$ with IOBE tags respectively. tag_t means the tag of the node having $preProb_t$ for $word_{i-1}$. In Step (5a), $HISTORY_t$, which has the previous tag feature of tag_t, is taken to compute the probability of $word_i$ with tag_j by the trained *MEmodel* described in Section 3.2. For example, if current tag_t is “B”, then $HISTORY_t$ should have an active feature “*Current-PreTag-Is-B*”. Then, the computed probability is multiplied by $preProb_t$ as in the principle of MEMM. *MEMMprob* and tag_t as the current previous node are stored in *tmpProbArr* (Step (5b)). After the probabilities of four different observation sets are computed, in Step (5c), we select the highest probability as the probability of current node *curProb*, and at the same time get the previous node that owns the best path *curPreTag*. Then, we store the probability of

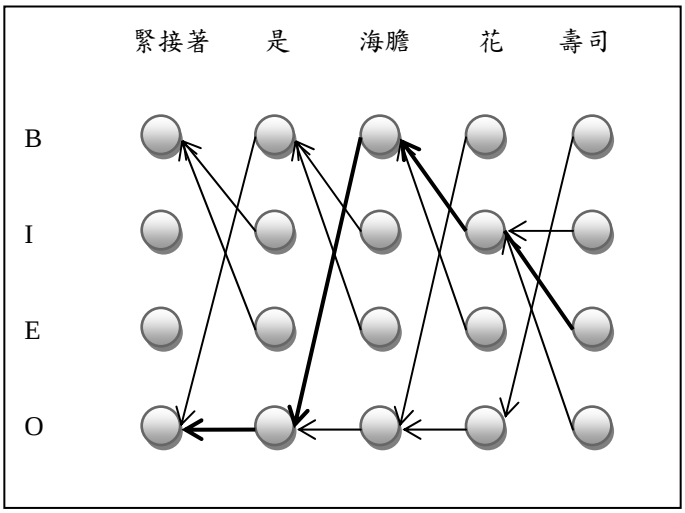


Figure 5: The *DPtable* of "緊接著是海膽花壽司" in which each node is connected to its best previous node and the bold lines are the best tagging sequence by Viterbi search

current node into *DPtable*. More formally, we can state the process of MEMM computation as:

$$MMpr_{\mathrm{word_i}}(tag_{\mathrm{j}}) = \max_{\mathrm{t=1}}^{4} MMpr_{word_{\mathrm{i-1}}}(tag_{\mathrm{t}}) * MEpr_{word_i}(tag_{\mathrm{j}}|HISTORY_{\mathrm{t}}) \quad (3)$$

where $V_{\mathrm{word_i}}(tag_{\mathrm{j}})$ is the MEMM probability of current node, $P_{word_i}(tag_{\mathrm{j}}|HISTORY_{\mathrm{t}})$ is the ME probability of current word $word_{\mathrm{i}}$ with current observation set $HISTORY_{\mathrm{t}}$, $V_{word_{\mathrm{i-1}}}(tag_{\mathrm{t}})$ means $preProb_{\mathrm{t}}$ in the algorithm of Figure 4. For *curPreTag*, we can state the process formally as below:

$$MMpr_{\mathrm{word_i}}(tag_{\mathrm{j}}) = \mathrm{argmax}_{\mathrm{t}}\, MMpr_{word_{\mathrm{i-1}}}(tag_{\mathrm{t}}) * MEpr_{word_i}(tag_{\mathrm{j}}|HISTORY_{\mathrm{t}}) \quad (4)$$

where *t* corresponds to *curPreTag*. *curPreTag* is store into *BackTrackTable*.

Once *DPtable* has stored the conditional probabilities of all the words and *BackTrackTable* has connected each node to its best previous node, we can decode a best tagging sequence backward by Viterbi search in Step (6). An example of finished *BackTrackTable* is shown in Figure 5, in which each node is connected to its best previous node and the bold lines indicate the best tagging sequence. Then, the target

instances can be extracted by the rules of BIEO chunks, as Figure 5 in which the target instances “海膽花壽司” can be extracted. An example of tagging sequence and extraction for an article on our working prototype is shown in Figure 1.

Chapter 4 Experiment Setting and Result

We carried out experiments to extract from a natural language text a set of instances in a more fine-grained type of a domain that lacks annotated data. As such, a machine learning model will be trained on an automatically tagged corpus and evaluated over a set of natural language texts containing target instances. Furthermore, we evaluate our method on the level of exact match; that is, a target instance must be exactly the same as the answer to be considered correct. In this chapter, we first present the setup of our training and test procedure for the evaluation (Section4.1). Then, Section 4.2 lists the systems that we developed for the comparison. The evaluation metrics for the performance of developed systems are introduced in Section 4.3. Finally, Section 4.4 reports and discusses the results of the evaluation using the methodology described in previous sections.

4.1 Experimental Setting

We select foods as our training and test type in the evaluation; that is, our experiment is for the extraction of eatable things as the target instances, such as dishes, drinks (e.g., “玉米麵餅”, “特調冰咖啡”). A set of 4,557 instances are selected as seeds to automatically collect and tag a corpus for training, obtained from three off-the-shelf thesauri, Chinese-English bilingual Wordnet, TonYiCi CiLin (同義詞詞林), and Chinese thesaurus provided by CKIP group of Academia Sinicia, all of which contain instances in various domains and categories. For Wordnet, we select the instances labeled as <noun.food>, indicating that an instance belongs to the food category. For TonYiCi CiLin, we select the food categories of Br01and Br03 to Br12.

Thesauri	Category	Number	Total	Total
Chinese Wordnet	noun.food	2461	2461	3155
TonYiCi CiLin	Br01 (糧食)	53	694	
	Br03 (食物)	15		
	Br04 (飯，粥，鍋巴)	33		
	Br05 (麵食)	50		
	Br06 (菜肴)	157		
	Br07 (調味品)	26		
	Br08 (油，塩，醬，醋，糖)	79		
	Br09 (點心)	80		
	Br10 (糖果)	23		
	Br11 (果料，餡料，酵子)	32		
	Br12 (飲料，乳酪)	146		

Table 6: The number of instances in the selected categories of two thesauri

Thesaurus	Semantic labels	Examples	Number	Total
Chinese thesaurus	food	零食	72	1402
	meals	水餃，牛肉	519	
	drinks	茶，奶，可樂	144	
	spices	冰糖，沙茶	89	
	fruits	芒果，香蕉	171	
	birds	雞，鴨	192	
	plants	大白菜	38	
	marine	魚，牡蠣	177	

Table 7: The number of instances with selected semantic labels from Chinese thesaurus provided by CKIP group of Academia Sinicia

The category Br02 contains instances of feed for domestic animals, so it is not selected. The number of instances in two thesauri is shown in Table 6. For Chinese thesaurus, we selected 1402 instances with the semantic labels indicating foods. The manual selection process required can be done in a few minutes. The details of selected labels and examples are listed in Table 7. More samples of the seed data used for evaluation are listed in appendix.

Parameter	Value	Description
minQLength	4	Minimum length of a seed for querying a search engine
*numSeed*s	1000	Number of seeds used to query a search engine for data collection
minCharCount	2	Minimum frequency for generating character-class lists
windowSize	2	The window size for preceding and subsequent features in the ME model

Table 8: Training parameters

Our experiments used a number of parameters in the training process. We performed some experimentation with the different values of these parameters resulting in the parameters shown in Table 8. We did not test parameters exhaustively and further fine-tuning may improve the performance of the system. As shown in Table 8, we randomly selected 1,000 instances with at least four characters and sent them to a search engine to collect data. We chose *Google* as the search engine because of its well known effective performance. A number of 40,535 snippets were retrieved and segmented into 1,994,222 words. The Chinese segmenter was done using an in-house segmentation system.

In the development stage of our method, we had a set of developing data, including ten articles of about 500 words. From the observation of developing data, we found, by checking part-of-speech of each word of seed data, some words usually do not belong to our target instances and thus should be excluded during the automatically tagging process; otherwise, they may create noise. For example, in the Chinese instances of food, quantifiers usually show up near the end of an instances, such as “麵包(Na) 捲(Nf)” or “包(Nf) 餡(Na) 卷(Nf)”, where “捲”, “包” and “卷” are quantifiers (Abbreviations inside parentheses are part-of-speech tags provided by our Chinese segmenter). However, when quantifiers show up in front of food instances, most of the time they indicate the amount of a noun and thus do not belong

Part-of-speech	**Examples**
Quantifier (Nf)	“麵包(Na) 捲(Nf)”, “包(Nf) 餡(Na) 卷(Nf)”
Conjunction (CA, CB, CC)	“博(bb) 若(CC) 來(Db) 葡萄酒(Na)”， “雜(VH) 和(CA) 菜(Na)”
Expletive (T)	“勃(bb) 良(bb) **地(Te)** 葡萄酒(Na)”
Alphabet	Italien
Adverb (Da, Db)	“**全(Da)** 麥(Na) 餅乾(Na)”, “塔(bb) **可(Db)** 餅(Na) 醬(bb)”

Table 9: Inappropriate words for the regular expression in automatic tagging

Part-of-speech	**Example**
Alphabet (Fw)	XO
Number (Fw)	1, 2, 3
Punctuation (?X)	, . ? !
Adverb (Da, Db)	也，總共，都
Conjunction (CA, CB, CC)	和，與
Postposition (NG)	以上，後，裡
Expletive (T)	地，啊，哦
Location (LG)	上面，側邊
(DE)	的，之，得，地
(VQ)	是
(VX)	有
(VHEcDaDF)	多
(NfDE)	些
(NfNd)	元，小時

Table 10: The part-of-speech filtered out in post-process

to our target instances, such as “一卷錄音帶” or “一包餅乾”. Therefore, we restrict that quantifiers cannot be *Btoken* but they can be in *Itoken* or *Etoken*. English words are also not considered because our method is designed for identifying Chinese NER. Other words that are not part of any named entity include conjunction (CA, CB, CC), expletive (T) and some adverbs (Da, Db). All the words are excluded using regular expression and the examples are listed in Table 9.

From the developing data, we also found that in the test procedure, additional

post-processing can be applied by checking part-of-speech of each word to filter out the tokens that are often wrongly tagged as parts of our target instances, maybe for the reason that their preceding or subcequent features elevate the probability as part of target instances. Recall that preceding and subsequent features are to include all the word-class, character-class, last character, bigram, part-of-speech, and semantic label features from the surrounding tokens at a certain window size into current token (In the experiment, the window size is two, as in Table 8). All the filtered part-of-speech are listed in Table 10 and applied in the experiment.

Testing data were obtained from 30 blog articles about users experiences of dining in some restaurants. We randomly selected these articles from food forum in the PTT bulletin board system (BBS). PTT, as the biggest BBS in Taiwan, contains various forums on diverse topics. There are approximately 100,000 users at any time on PTT. Food forum at PTT is an active area for users to discuss and share their experiences with restaurants; therefore, it is easy for us to obtain articles containing our target instances. Human annotation was done on these articles for the evaluation of exact match. Criteria of annotation will be described in Section 4.3.

4.2 Systems for comparison

Our experimental evaluation focuses on how a trained ME model can improve accuracy of extraction of target instances. We compare the result with a baseline evaluation, the data-driven system, which is the automatic tagging approach in the training process, to see if we can improve the performance. Then, we train the ME model using the methodology described in Chapter 3 for the extraction of food instances. Recall that in the ME model, we use word-class, character-class, last character, bigram, part-of-speech, semantic labels, preceding and subsequent, and

previous state features. The parameters for these features are listed in Table 8. Now, we state the systems evaluated as follow:

- Data-driven: The method of automatic tagging using the regular expression of seed data described in Section 4.1.
- ME using *thesauri*: Using the seed data (described in Section 4.1), the trained ME model and MEMM procedure (described in Chapter 3) to extract target instances.

Recall that in Section 3.2.2, we discuss the benefit of the word-class, character-class, last character and bigram feature lists is that they allow us to incorporate different knowledge sources to expand the amount of data in lists. Therefore, we expand these feature lists of ME using *thesauri* system by adding instances obtained from a web forum called "非凡大探索" (Unique Discovery) (http://www.ustv.com.tw/viewforum.php?f=33), which is a forum of a TV show in Unique Satellite TV Channel (非凡電視台) introducing delicacies found in Taiwan. In the Unique Discovery Forum, the discussions from 2005/12/31 to 2008/08/17 were selected and compiled into a list of instances *UDList* mainly about Taiwanese food. We gathered more seed data from this Web forum with the goal of showing the simplicity and effectiveness of our training procedure since collecting seeds from the Web is much easier than the preparation of annotated corpus. We will compare the performances about different amount of *UDList* added in the feature lists. We also compare the performances with and without word-class as well as character-class features to observe whether they help cope with data sparseness and improve performance. Additionally, the preceding and subsequent features (Uchimoto et al., 2000) are also examined for effectiveness.

4.3 Evaluation Metrics and Annotation on Test Data

Recall that our approach starts with a natural language text created by a user, and reduces the text into a tagging sequence of BIEO. The output of our approach is a set of instances of a certain type in a domain, which can either be shown to the user directly, or used as the index of a vertical search engine, the answers of a question answering system. It is crucial that a system can extract fully correct instances, so the index or answers will be meaningful for users. Therefore, our evaluation is a simple scoring protocol the same as IREX and CONLL, "exact-match evaluation".

Information extraction systems are usually compared based on the quality of the extracted instances. This quality is traditionally quantified using three metrics, *precision*, *recall*, and *F-measure*. *Precision* is calculated as the fraction of exactly correct instances among all the instances extracted, and *recall* measures the coverage of the system as the fraction of the real answers in the test data that are successfully extracted by the system. *F-measure* considers both *precision* and *recall* of the test data to compute a score that can measure a test's accuracy, as the harmonic mean of *precision* and *recall*.

To evaluate our approach, we inspect the instances extracted by the various systems that we compared (Section 4.2). We will describe how we annotate instances of food instances in the test data later. Using the annotated test data, we evaluate the instances that the systems extract using the *precision*, *recall*, and *F-measure*.

Definition 4.1. The *precision* of an extracting system *Sys* for test data is the percentage of correct instances among all the instances returned by *Sys*.

EXAMPLE 1. *Consider 1,000 instances extracted by Sys from test data. If 700 of these 1000 instances are fully correct with the answers, then the precision is* $\frac{700}{1000} * 100\% = 70\%$.

Definition 4.2. The *recall* of *Sys* is the percentage of correct instances among all the answers of test data.

EXAMPLE 2. *Consider test data contain 900 answers. If 700 of the 1000 instances extracted by Sys are fully correct with the answers, then the recall is* $\frac{700}{900} * 100\% = 77\%$.

Definition 4.3. The *F-measure* of *Sys* is the mean of precision and recall. The formula is as follow:

$$Fmeasure = \frac{2*(precision * recall)}{(precision+recall)} \quad (4)$$

EXAMPLE 3. *If the precision is 70% and the recall* is *77%, then the F-measure is* $\frac{2*(0.7*0.77)}{(0.7+0.77)} = 0.72$.

Once we have trained a ME model for food instances as discuss in the previous sections, we evaluate the performance of the extracting systems in Section 4.2 using the evaluation metrics. We annotate the test data to mark the food instances in the text as the answers. The definition of food instances is eatable things or dishes that are usually seen on menus, such as “牛肉麵”, “焗烤蔬菜飯”; additionally, more general instances, such as “菜”, “飯”, “肉”, or “豬肉” are also marked. Note that if two or more food instances are connected together, we consider them as one instances, such as “水果鰻魚壽司” because in Chinese syntax, when a noun locate right before another noun, the first noun is usually used as the modifier of second noun to make them as a multi-words unit. On the other hand, if there is a conjunction, preposition or punctuation between two food instances, we treat them as two separate instances, as the example that “水果和壽司” have two instances “水果” and “壽司”.

In the evaluation, we only consider those instances that are three characters or longer because in our empirical observation, instances composed by less than two characters are often general terms and can be simply looked up in dictionaries.

However, instances of three characters or more are more likely named entities, which are often not included in dictionaries. After human annotation, there are totally 823 food instances in the test data. By checking Chinese Wordnet and Chinese thesaurus provided by Academia Sinicia, only about 13% of answers containing three characters above are in the dictionaries.

4.4. Evaluation Results

In this section we report and discuss the results of the experimental evaluation using the methodology described in Chapter 3. First, we report the performances of two systems, data-driven and ME using *thesauri* on the test data. Then, we compare the results of different amount of *UDList* adding into ME using *thesauri*. Finally, the significance of word-class and character-class features is discussed by comparing the results of ME using *thesauri* and ME using *thesauri* plus *UDList* with and without these features. Additionally, we also examine the effectiveness of the preceding and subsequent features (Uchimoto et al., 2000).

During this evaluation, 30 articles were through the run-time procedure described in Section 3.3 to extract target instances, foods. Table 11 shows the *precision*, *recall*, and *F-measure* of two systems: data-driven and ME using *thesauri*. As we can see, the *precision* rate of the data-driven system is about 0.66. About two thirds of correct instances suggest that our automatically tagging approach can correctly match target instances to a certain degree. On the contrary, the *recall* of data-driven system is relative low maybe due to the reason that using the criteria of exact match for an instance, such the instance segmented as “涼拌 | 魚膠”, the data-driven system can not extract the unknown word “魚膠” even though its components (characters) and previous word are the obvious characteristics of food

	Precision	Recall	F-measure
Data-driven	0.66 (305 / 458)	0.37 (305 / 823)	0.47
ME using *thesauri*	0.70 (452 / 639)	0.54 (452 / 823)	0.61

Table 11: The results of two systems: data-driven and ME

instances. Thereby the *F-measure* is less than 0.5. However, we believe that in our training data, the *recall* may be higher for the reason that since the snippets are retrieved by the seed data, the possibility of unknown words may be lower.

In Table 11, the ME using *thesauri* shows improvement on both *precision* and *recall*. The *precision* rate is about 0.70, indicating that after applying machine learning on the automatically tagged data, the MEMM procedure can increase the *precision* rate by using the additional features. Also note that the ME using *thesauri* substantially improves the *recall*, showing that the machine learning model solves the problem of data sparseness to some extent, as the example that the instance "涼拌魚膠" can be correctly extracted since the characters of "魚膠" are commonly used elements in foods and "涼拌" is often the *Btoken* of foods. The *F-measure* of ME using *thesauri* increases to over 0.6, outperforming that of data-driven system. This improvement indicates that the instances derived from thesauri already contain some commonly used elements and general terms, such as "飯", "麵", "牛排", "麵包", "三明治". The problem of data-sparseness can also be improved by our discovered features.

Recall that we can expand the word-class, character-class, last character and bigram list features of the ME using *thesauri* system by adding instances from *UDList* (described in Section 4.2), which contains a lot of Taiwanese dishes, to see the influence of increasing amount of data. Table 12 shows the comparison of ME using *thesauri* and ME using *thesauri* plus *UDList*. As we can see, the ME using *thesauri* plus *UDList* shows a great improvement on *precision* and *recall*. The *precision* is

	Precision	Recall	F-measure
ME using *thesauri*	0.70 (452 / 639)	0.54 (452 / 823)	0.61
ME using *thesauri* + *UDList*	0.78 (615 / 788)	0.74 (615 / 823)	0.76

Table 12: The comparison of ME and ME using *UDList*

increase drastically to 0.78 while the *recall* to 0.74, which are significant improvement over the ME using *thesauri*. This improvement may be due to the fact that TonYiCi CiLin and Chinese thesaurus contain mostly the general terms, such as "麵粉" or "飯", which can not cover enough commonly used elements for food instances. Chinese Wordnet is basically the translation version of English Wordnet, and thus may not include a lot of instances for Chinese food. Therefore, it is likely that the elements of *Btoken* are not enough and our discovered features can not solve all the problem of unknown *Btoken*, so adding *UDList* let the model can learn more *Btoken* to correctly extract foods. For example, *Btoken* like "日式", "和風", "麻辣", or "手工" commonly shows up in Taiwanese food instances but is not included in the thesauri. Adding *UDList* allow the ME model to extract instances like "日式南瓜鍋".

Additionally, we examined whether *UDList* can actually replace thesauri all together, so we remove the data of thesauri from the word-class, character-class, last character and bigram list features, to see if the ME model only using *UDList* can achieve the same performance as the same model using *thesauri* plus *UDList*. Table 13 shows the results of ME only using *UDList* and ME using *thesauri* plus *UDList*. As Table 13 shows, although the *recall* rate stays almost the same, the *precision* is greatly decreased by 0.10, even lower than the ME using *thesauri*, indicating that without thesauri a lot of noise is created. The reason may be that although *UDList* contains many elements of Taiwanese foods, some may not be important elements and their importance is falsely elevated without the reinforcement of data from thesauri, especially for *Itoken* and *Etoken*. As for the ME using *thesauri*, thesauri may

	Precision	Recall	F-measure
ME using *thesauri*	0.70 (452 / 639)	0.54 (452 / 823)	0.61
ME using *UDList*	0.69 (617 / 822)	0.74 (617 / 823)	0.72
ME using *thesauri* + *UDList*	0.78 (615 / 788)	0.74 (615 / 823)	0.76

Table 13: The results of ME using *thesauri*, ME using *UDList* and ME using thesauri + *UDList*

contain a lot of general terms but not enough elements for *Btoken* or *Itoken*, so some answers may be partially extracted such that *recall* is relatively lower. Therefore, ME using *thesauri* plus *UDList*, as a combined system, can learn the advantages from both ME using *thesauri* and ME using *UDList* to present a better performance. As the example segmented sentence "味道 | 完全 | 蓋過 | 煙燻 | 鮭魚 | 滋味", ME using *thesauri* plus *UDList* can correctly extract "煙燻鮭魚", while ME using *thesauri* and ME using *UDList* wrongly extract "煙燻鮭魚滋味". The reason may be that the part-of-speech of "滋味" is a common noun (Na), which often shows up as an *Itoken* or *Etoken* in Chinese foods (e.g., 美奶"滋" in Chinese Wordnet, 川"味"小炒 in *UDList*), and some of its characters are in *IcharList* and *EcharList* of *UDList*, so according to the part-of-speech and the interchangeability of *Itoken* as well as *Etoken* the model decides "滋味" to be *Etoken* of the instance. However, using thesauri plus *UDList*, which contain many fundamental instances, especially *Etoken*, as well as *Btoken* for Taiwanese food allows the model to learn when the current characters are in *IcharList* and its previous word is an *Etoken*, the current word is not likely to be an *Etoken*.

We also investigated whether or not the performance depends on the number of instances in *UDList*. There are totally 2,932 food instances in *UDList*. First, we randomly select three groups of instances from *UDList*, named *UDList-1000-a*, *UDList-1000-b*, and *UDList-1000-c*, each of which contain 1,000 instances, in order to see if adding the same amount of three groups with different members from *UDList*

	Precision	Recall	F-measure
ME using *thesauri + UDList-1000-a*	0.75 (554 / 737)	0.67 (554 / 823)	0.71
ME using *thesauri + UD List-1000-b*	0.76 (558 / 730)	0.67 (558 / 823)	0.71
ME using *thesauri + UD List-1000-c*	0.76 (565 / 740)	0.68 (565 / 823)	0.72

Table 14: The results of adding UDList-1000-a, UDList-1000-b, UDList-1000-c to the ME using thesauri

respectively to the ME using *thesauri* system can achieve similar performance. Table 14 shows the results of adding the three groups respectively to the ME using *thesauri* system. As we can see, the *precision* and *recall* of three groups are approximately the same. Comparing to the ME system using full *UDList* in Table 12, the *precision* only decrease about 0.02, suggesting that once the feature lists contains some elements of Taiwanese food, our machine learning model can identify food in Chinese text more effectively. The reason may be that the missing elements of original ME using *thesauri* system is mainly *Btoken* in Taiwanese foods. These *Btoken* are limited and can be acquired in small amount of data. On the other hand, the *recall* decreases about 0.06, indicating that amount of elements of Taiwanese food still limit the coverage of the ME system.

To examine variations of *recall* more specifically, we evaluate the performances of changing the amount of instances from *UDList* added to the ME system. *UDList-1000-a* in Table 11 is taken as the group with the least amount of instances from *UDlist*, *UDList-1000*, since it has the weakest performance in Table 14. Upon *UDList-1000*, we randomly select 1000 instances, which do not exist in the instances of *UDList-1000*, from *UDList*, to make a list of 2000 instances, *UDList-2000*. Finally, ME using *thesauri* plus full *UDList* is also put into this comparison since there only about 3000 instances in *UDList*. Thereby we increase 1000 instances at each group to see the effects with different amount of Taiwanese food instances. The results are

	Precision	Recall	F-measure
ME using *thesauri + UDList-1000*	0.75 (554 / 737)	0.67 (554 / 823)	0.71
ME using *thesauri + UDList-2000*	0.76 (583 / 752)	0.70 (583 / 823)	0.74
ME using *thesauri + UDlist*	0.78 (615 / 788)	0.74 (615 / 823)	0.76

Table 15: The results of ME using *thesauri* plus *UDList-1000*, *UDList-2000*, and *UDList* repectively

shown in Table 15.

As indicated in Table 15, with increasing amount of instances, the *recall* rates are stably improved, about 0.04 increased at each group. This shows that the coverage of ME model is based on the number of instances in the feature lists. However, the *precision* rate stays more or less the same. The ME using *thesauri + UDList-2000* even have similar precision as that using *thesauri + UDList-1000-b* or *UDList-1000-c*, indicating that most of the Taiwanese food contain about the same elements, which can be acquired from a small amount of Taiwanese food instances, such as "日式", "法式", "義式", "和風" and so on. Once our model learns these elements, the system can identify Taiwanese food more correctly using the designated features.

Finally, we examine the relation between the performances and individual feature. We found that the word-class and character-class features are the most important. Table 16 shows the results of the ME using *thesauri* and the ME using *thesauri* plus *UDList* when word-class and character-class features were removed. First, we remove both word-class and character-class features, and the results show that the *precision* does not change much, especially for the ME using *thesauri* whose *precision* almost stays the same, maybe because the bigram list and semantic labels features already provide enough information, which can correctly identity food instances. However, without two class-based features, the *recall* is obviously much lower, suggesting that missing class-based features prevent the machine learning

	ME using *thesauri*			**ME using *thesauri* + *UDList***		
	Precision	Recall	F-measure	Precision	Recall	F-measure
- Word-class - Character-class	0.70 (283 / 404)	0.34 (283 / 823)	0.46	0.75 (420 / 554)	0.51 (420 / 823)	0.61
- Character-class	0.70 (427 / 609)	0.51 (427 / 823)	0.59	0.75 (603 / 798)	0.73 (603 / 823)	0.74
+ All features	0.70 (452 / 639)	0.54 (452 / 823)	0.61	0.78 (615 / 788)	0.74 (615 / 823)	0.76

Table 16: The results of ME and ME-plus systems with or without word-class and character-class features

model from recognizing more food instances. When we only removed the character-class features, the *recall* was greatly improved, suggesting that word-class features, as categorized unigrams, provide significant characteristics for the model. However, the *precision* still stay the same, meaning that word-class features can improve the coverage but can not increase the degree of exactness. When all the features, including character-class and word-class features, are used, the *recall* rate increases slightly and more importantly, the precision of ME using *thesauri* plus *UDList* was improved. It is interesting that the *precision* of the ME using *thesauri* plus *UDList* increases while the ME using *thesauri* does not. It is possible that *UDList* contains various elements which can let the model extract more instances by their significant character-class features and at the same filter out some noise when they do not have suitable character-class. As for the ME using *thesauri*, the character-class features seem not to contain diverse elements enough, especially for the elements of Taiwanese foods, thereby unable to extract as more instances as the ME using *thesauri* plus *UDList* does to improve the accuracy. For example, the ME using *thesauri* can only recognize partial answer “蕃茄培根義大利麵” while ME using *thesauri* plus *UDList* can extract the full answer “辣味野菇蕃茄培根義大利麵”. By

Feature set	ME using *thesauri*			ME using *thesauri* + *UDList*		
	Precision	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
(0)	0.67 (241 / 359)	0.29 (241 / 823)	0.40	0.77 (399 / 513)	0.48 (399 / 823)	0.59
(-1) to (1)	0.70 (431 / 609)	0.52 (431 / 823)	0.60	0.74 (465 / 625)	0.56 (465/ 823)	0.64
(-2) to (2)	0.70 (452 / 639)	0.54 (452 / 823)	0.61	0.78 (615 / 788)	0.74 (615 / 823)	0.76
(-3) to (3)	0.68 (459 / 667)	0.55 (459 / 823)	0.61	0.73 (606 / 820)	0.73 (606 / 823)	0.73

Table 17: The results of features sets of current token plus those of different surrounding tokens

using character-class features with the information in *UDList*, the model learns "辣味" as a beginning token, and "野" as well as "菇" as inside tokens.

We further examine the performance of using surrounding features (preceding and subsequent features). Recall that in the previous experiment we all use the window size of two for the preceding and subsequent features. In Table 17, we test different values of window size to see if the results are consistent with Uchimoto's work. In Table 17, "(0)" means that only the features from the current token were used. "(-1) to (1)" indicates that we used features from the current token and its two adjacent tokens. "(-2) to (-2)" indicates that we used features from the current token and its four adjacent tokens, the two on the left and the two on the right of the current token. "(-3) to (3) indicates that we used features from the current token and the six nearest tokes, the three on the left and the three on the right. As we can see, for "(0)", the *precision* of both ME using *thesauri* and ME using *thesauri* plus *UDList* systems are slightly decrease, while their *recall* are greatly decreased, suggesting that although our discovered features already can handle a certain degree of extraction, at the same time they can not deal with data sparseness by surrounding features, and thus the coverage of the extraction becomes insufficient.

For the “(-1) to (1)” case of Table 17, the *recall* is significantly increased, meaning that the problem of coverage is improved by adding surrounding features. However, it is interesting that the *precision* of ME using *thesauri* plus *UDList* is 0.03 lower than that of “(0)”, possibly for the reason that using features of two adjacent tokens still does not have enough information to deal with the problem of data sparseness. At the same time, it may introduce the noise because of the diverse elements of *UDList*. As the example sentence segmented as “這 | 道 | 絲瓜 | 完全 | 是 | 品嘗 | 原味”, the ME using *thesauri* plus *UDList* of the “(0)” case can successfully extract “絲瓜”, while ME using *thesauri* plus *UDList* of “(-1) to (1)” extracts “道絲瓜” because the probability of “道” as a part of a food instance is raised by the surrounding features from “絲瓜”, and “道” is one of beginning tokens in *UDList* (e.g., “道口燒雞” in *UDList*).

For the “(-2) to (2)” case in Table 17, the *precision* and *recall* of ME using *thesauri* plus *UDList* are substantially improved, probably for reason that more surrounding information can allow the model to automatically learn more specific rules for the diverse elements of *UDList*. As the example of “道絲瓜”, the ME using *thesauri* plus *UDList* of “On (-2) to (2)” can let “絲瓜” successfully recognize that “道” should not be its *Btoken* in this sentence by its surrounding token “這” and “是”. On the other hand, the ME using *thesauri* of “(-1) to (1)” and “(-2) to (2)” have almost the same *precision*, suggesting that since without the diverse elements of *UDList*, the both ME systems can achieve similar performance by using surrounding features.

Finally, for the “(-3) to (3)” case of the ME using *thesauri*, Table 17 shows that the *precision* is slightly decreased while the *recall* is slight increased; therefore, the performance stay almost the same. However, the performance of ME using *thesauri* plus *UDList* with “On (-3) to (3)” is obviously decreased. The reason might be that the model may learn too specific rules since the ME using *thesauri* plus *UDList*

contains so many diverse elements. As the example sentence “而 | 是 | 它 | 雲 | 醬 | 高麗菜 | 啊”, the ME using *UDList* of “(-2) to (2)” successfully extracts “雲醬高麗菜”, while that of “(-3) to (3)” extracts “醬高麗菜” because the previous tokens “而”, “是” and “它” may lower the possibility of “雲” as a *Btoken*, and “雲” and “醬” are both in *BtokenList* and *ItokenList*, such that the model may learn too specific rules preventing “雲” from being a part of the food instance.

Therefore, from the discussion above, our results confirm the discovery of Uchimoto et al. (2000) that the best accuracy was achieved by using the features of current token and the four nearest tokens.

Chapter 5 Conclusion and Future Work

In summary, we have introduced a semi-supervised method for the extraction of instances of a certain type in a domain by a machine learning model developed from an automatically collected and tagged corpus. The method involves selecting seed data of target instances from off-the-shelf general purpose thesauri, using seeds to automatically collect a corpus from the Web, automatically tagging the corpus by seed data and training a machine learning model on the corpus. We have implemented and thoroughly evaluated the method as applied to natural language texts. In the evaluation of exact match, we have shown that the method successfully extracts target instances that meet a certain degree of accuracy, about 78%, by using a small amount of seed data and automatic procedure for the constitution of training data. Our proposed method has the advantage of using less expensive development.

Many avenues exist for future research and improvement of our system. From our examination of results, the problem of data sparseness still exists; therefore, more domain knowledge or heuristics could be used to improve the identification of target instances. More effective features for the ME model could be discovered in further experiments. For example, more class-based features could be applied for improving the performance of the given target instances. Since we successfully use small amount of seeds to develop an IE system, bootstrapping strategy of repetitive crawling, using more and more instances, may allow using a smaller amount of data to build a very large corpus. Additionally, an interesting direction to explore is to perform rule-based post-processing for creating a hybrid system, for the reason that we find a lot of unknown words are brands of products, such “黑炫” in the instance “黑炫巧克力餅乾”, from which our method can only extract “巧克力餅乾”. Using more

sophisticated post-processing may solve the problem of unknown words to some extent.

References

Bender, O., Och, F. J., & Hermann, N. (2003). Maximum Entropy Models for Named Entity Recognition. *In Proceedings of the seventh conference on Natural language learning at HLT-NAACL.*

Borthwick, A., Sterling, J., Agichtein, E., & Grishman, R. (1998). NYU: Describption of the MENE Named Entity System as used in MUC-7. *In Proceedings of the Seventh Message Understanding Conference (MUC-7).*

Brin, S. (1998). Extracting Patterns and Relations from the World Wide Web. *In WebDB Wokshop at EDBT '98.*

Downey, D., Broadhear, M., & Etzioni, O. (2007). Locating Complex Named Entities in Web Text. *In Proceedings of IJCAI 2007.*

Hearst, M. (1992). Automatic Acquisition of Hyponyms from Large Text Corpora. *In Proceedings of the 14th conference on Computational linguistics.*

Kripke, S. *Naming and Necessity.* Boston: Harvard University Press.

Lisa, R. F. (1991). Extracting Company Names from Text. *In Proceedings of Conference on Artificial Intelligence Applications of IEEE.*

Manning, C. D., Raghavan, P., & Schutze, H. (2009). *Introduction to Information Retrieval.* Cambridge university press.

McCallum, A., Freitag, D., & Pereira, F. (2000). Maximumentropy Markov models for information extraction andsegmentation. *In Proceedings of ICML* (pp. 591–598). California: Stan-ford.

Nadeau, D., & Sekine, S. (2007). A Survey of Named Entity Recognition and Classification. *Lingvisticae Investigationes, Volume 30, Number 1* , pp. 3-26.

Nadeau, D., Turney, P. D., & Matwin, S. (2006). Unsupervised NamedEntity Recognition: Generating Gazetteers and Resolving Ambiguity. *In Proceedings of Canadian Conference onArtificial Intelligence.*

Patwardhan, S., & Riloff, E. (2006). Learning Domain-specific Information Extraction Patterns from the Web. *In Proceedings of the ACL 2006 Workshop on Information Extraction Beyond the Document.*

Riloff, E., & Jones, R. (1999). Learning Dictionaries for Information Extraction by Multi-level Bootstrapping. *In Proceedings of the Sixteenth National Conference on Artificial Intelligence.*

Sekine, S., & Isahara, H. IREX: IR and IE Evaluation project in Japanese. *In Proceedings of the 2nd International Conference on Language Resources and Evaluation.*

Sekine, S., Sudo, K., & Nobata, C. (2002). Extended Named Entity Hierarchy. *In Proceedings of the LREC-2002 Conference.*

Shinzato, K., Sekine, S., Yoshinaga, N., & Torisawa, K. (2006). Constructing Dictionaries for Named Entity Recognition on Specific Domains from the Web. *In Web Content Mining with Human Language Technolo-gies Workshop on the 5th International Semantic Web.*

Tjong Kim Sang, E. F., & De Meulder, F. (2003). Introduction to the CoNLL-2003 Shared Task:Language-Independent Named Entity Recognition. *In Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003.*

Tsai, T.-H., Wu, S.-H., Lee, C.-W., Shih, C.-W., & Hsu, W.-L. (2004). Mencius: A Chinese Named Entity Recognizer Using Maximum Entropy-based Hybrid Model. *International Journal of Computational Linguistics & Chinese Language Processing* .

Uchimoto, K., Ma, Q., Murata, M., Ozaku, H., & Hitoshi, I. (2000). Named Entity Extraction Based on A Maximum Entropy Model and Transformation Rules. *In Proceedings of 33rd Annual Meeting of the Association of the Computational Linguistics.*

Zhang, Y.-J., & Zhang, T. (2007). ME-based Chinese Person Name and Location Name Recognition Model. *In Proceedings of the Sixth International Conference on Machine Learning and Cybernetics.*

Zhao, J., & Liu, F. (2008). Product Named Entity Recognition in Chinese Text. *Language Resources and Evaluation* .

Appendix A – Samples of Seed data

Source: Chinese Wordnet

- 丁香
- 丁骨牛排
- 人心果樹
- 八角茴香果
- 十字花科蔬菜
- 十字霜糖麵包
- 十字麵包
- 三丁煎蛋捲
- 三明治
- 三明治餐
- 三層三明治
- 下午茶
- 上等牛排
- 上等紅茶
- 上腰肉牛排
- 千島醬
- 千層麵
- 口香糖
- 口香糖球
- 土耳其式開胃菜
- 土耳其肉飯
- 土耳其咖啡
- 土耳其飯
- 大比目魚肉
- 大白菜
- 大肉塊
- 大西洋鮭魚
- 大西洋蟹
- 大豆
- 大豆油
- 大空心粉筒
- 大青豆
- 大型三明治
- 大紅香腸
- 大胡桃
- 大香腸
- 大茴香
- 大茴香果實
- 大茴香烈酒
- 大茴香餅
- 大梭魚肉
- 大理石蛋糕
- 大粒葡萄
- 大麥
- 大麥芽威士忌
- 大麥茶
- 大麥湯
- 大麥糖
- 大湖鱒魚
- 大黃
- 大黃派
- 大腿肉
- 大蒜
- 大蒜奶油
- 大蒜麵包
- 大嘴鱸
- 大蝦
- 大蕉
- 大餐
- 大鍋湯
- 大雜燴
- 大麗花
- 大蘋果
- 小口黑鱸
- 小牛肉
- 小牛肉片
- 小牛肉塊
- 小牛舌
- 小牛腦
- 小白菜
- 小羊肉
- 小西餅
- 小西點
- 小杏仁餅乾
- 小豆蔻
- 小型麵包

Source: ToYiChi CiLin (同義詞詞典)

- 糧食
- 食糧
- 糧
- 乾糧
- 侯糧
- 糗
- 粗糧
- 糙糧
- 米
- 大米
- 白米
- 秈米
- 機米
- 陳米
- 老米
- 碎米
- 粞
- 江米
- 糯米
- 糙米
- 糲
- 高粱米
- 秫米
- 米粒
- 糝
- 米粉
- 米面
- 麵粉
- 白麵
- 麵
- 玉米麵
- 棒子麵
- 麩子
- 麩皮
- 細糧
- 雜糧
- 原糧
- 口糧
- 軍糧
- 公糧
- 錢糧
- 餘糧
- 機動糧
- 商品糧
- 救濟糧
- 漕糧
- 粳米
- 黃米
- 炒米
- 小米
- 小米麵
- 豆麵
- 高粱麵
- 食物
- 食品
- 食
- 吃食
- 飯食
- 飯菜
- 熟食
- 煙火食
- 煙火
- 副食品
- 軟食
- 麵食
- 葷食
- 素食
- 零食
- 飯
- 米飯
- 白飯
- 白玉
- 白米飯
- 乾飯
- 蓋飯
- 蓋澆飯
- 份兒飯
- 客飯
- 剩飯
- 現飯
- 粗飯
- 糙米飯
- 脫粟飯
- 撈飯
- 齋飯
- 夾生飯
- 菜飯
- 八寶飯
- 粥
- 稀飯
- 米湯
- 糜
- 大米粥
- 臘八粥
- 魚生粥
- 雞粥
- 糖粥
- 赤豆粥
- 綠豆粥
- 鍋巴
- 鐊焦
- 麵食
- 麵條
- 麵
- 湯麵
- 麵湯
- 抻麵
- 拉麵
- 削麵
- 刀削麵
- 掛麵
- 卷麵
- 卷子麵
- 卷子
- 卷
- 饅頭
- 包子
- 饅首
- 饃
- 饃饃
- 餑餑
- 麵餅
- 餅子
- 餑餑
- 過水麵
- 雜麵
- 雜和麵兒
- 切麵
- 燙麵
- 冷麵
- 炒麵
- 拌麵
- 壽麵
- 麵坯兒
- 通心粉
- 死麵

Source: Chinese thesaurus provided by Academia Sinica

- 子
- 奶
- 瓜
- 冰
- 米
- 派
- 苗
- 凍
- 茶
- 蚌
- 酒
- 乾
- 梨
- 梟
- 蚵
- 雀
- 魚
- 鳥
- 湯
- 粟
- 粥
- 菸
- 雁
- 飯
- 蜆
- 飴
- 鳩
- 蜜
- 餅
- 餌
- 鳶
- 魟
- 蔥
- 蝦
- 醋
- 醅
- 魴
- 魨
- 橙
- 燕
- 糕
- 糖
- 餐
- 餡
- 鴨
- 鴛
- 龜
- 鮀
- 鮓
- 薑
- 醣
- 鮫
- 鮪
- 鴿
- 醃
- 鮚
- 鴳
- 黿
- 糧
- 雞
- 鯉
- 鵝
- 鵠
- 羹
- 蟹
- 鯨
- 鯛
- 鶉
- 鵲
- 鵰
- 麵
- 蠙
- 鰋
- 鰈
- 鯷
- 鶴
- 饘
- 鰣
- 鶡
- 鱉
- 鰻
- 鱗
- 鰲
- 鷲
- 鷹
- 鷺
- 鹽
- 鱣
- 鸇
- 鸎
- 鸞
- 丁香
- 九孔
- 人魚
- 八珍
- 八哥
- 八寶
- 三鮮
- 丸子
- 兀鷹
- 千鳥
- 口糧
- 土司
- 土豆
- 土雞
- 大豆
- 大菜
- 大雁
- 子規
- 小吃
- 小食
- 小鳥
- 小菜
- 山雀
- 山楂
- 山鵲
- 山鷂
- 川菜
- 川鹽
- 干貝
- 中飯
- 中餐
- 五香
- 五穀
- 介蟲
- 切麵
- 刈包
- 午飯
- 午餐
- 天鵝
- 孔雀
- 文旦
- 方糖
- 月餅
- 木瓜
- 毛豆

Source: *UDList*

- JUMBO 大漢堡
- Sophia 芝麻牛蒡
- XO 蝦醬麵
- XO 醬炒曼波魚
- XO 醬炒帶子
- XO 醬爆蚶頭
- xo 醬潤餅捲
- xo 醬鴨胸
- 一窩絲餅
- 一串心
- 一品沙鍋
- 一品鍋
- 一窩絲
- 七色炸物
- 七里香
- 七味雞柳
- 九孔燉雞
- 九州豚骨拉麵
- 九尾雞
- 九層塔牛肉絲
- 九層塔炒蟹
- 九層粿
- 九層粿加蛋
- 九環醬燒牛肉
- 九點米豬腳鍋
- 九轉肥腸
- 二節子麵
- 二層肉
- 人蔘花膠燉湯
- 人蔘雞湯
- 八仙湯
- 八仙飯
- 八寶冰
- 八寶豆花
- 八寶乾麵疙瘩
- 八寶湯
- 八寶菜
- 八寶釀雞翅
- 刀切豬腳麵
- 刀削牛肉麵
- 刀削炒麵
- 刁大根
- 十全燉物雞
- 十穀土司
- 卜肉
- 卜鴨
- 三口雞汁拌麵
- 三色布丁豆花
- 三色壽司
- 三色慕思
- 三色丼
- 三角湯圓
- 三角餅
- 三味雞
- 三味鱸魚
- 三明治
- 三杯土雞
- 三杯中卷
- 三杯地瓜
- 三杯花枝串
- 三杯花枝標
- 三杯花膠
- 三杯豬血糕
- 三杯豬尾巴
- 三杯雞
- 三杯雞米漢堡
- 三杯雞套餐
- 三絲丸
- 三樣冰
- 三樣湯
- 三鮮炒貓耳朵
- 三鮮拼盤
- 三鮮燴飯
- 三寶飯
- 下午茶套餐
- 下水湯
- 下水雞佛清湯
- 下水麵線
- 上海炸醬麵
- 上海菜飯
- 上海灌湯包
- 上等牛五花
- 上等牛舌
- 丸仔湯
- 丸蛋米苔目
- 丸類拼盤
- 千層油糕
- 千層酥
- 叉燒包
- 叉燒拉麵
- 叉燒豚骨拉麵
- 叉燒酥
- 口水雞
- 口袋燒餅
- 土司半條
- 土豆粽
- 土豆豬腳湯
- 土鳳梨酥
- 土雞爪
- 土鵝肉
- 土魠魚羹
- 大包子

Appendix B – Samples of Test Data

NO.	Text	Answers
1	在桌邊把木造圓形餐台撐開，端菜的侍者則右手單手成 90°的托著大餐盤過來，單膝著地的把大餐盤放到餐台上，然後再分送各人的食物，我想，在這裡送錯餐這種事是不在茹絲葵字典上的。 餐前酒：微酸的氣泡香檳酒 帶著香氣的氣泡香檳酒，舉起酒杯對著燈光照射時，還可看到細微的氣泡從底部慢慢升起，啜了一小口，嚐起來微酸，酒精度還好。 生煎日式鮪魚片 用紗袋包住的 Lemon 生煎日式鮪魚片一放上桌，就立刻吸引了目光焦點。 片成約 0.5cm 的生煎鮪魚，前後交疊的浸在淺淺的橙色醬汁裡，放在黑色瓷盤中間，旁邊就是用紗袋包住的 Lemon 和薑片、蘿蔔絲，還有一隻小鐵夾 鮪魚是先將上皮略煎過，再灑上些胡椒粒，在尚未擠上 lemon juice 時，我就先夾起一塊，蘸了些醬汁送入口中，鮪魚細嫩的口感跟牛肉極像，sauce 是芥末調成的醬汁，有芥末微辛的口感但不嗆，不會破壞了鮪魚的口感。再透過餐廳貼心以紗袋包住的 lemon，擠上幾滴 Lemon juice，嚐起來炎點微酸的口感又是一種層次了。 凱散沙拉 上桌時，羅蔓生菜已經被切成小口的份量，除了灑的細微的起士粉外，上面還鋪著切成片狀的白色薄起士片，還有油炸過還有溫度的麵包丁，吃起來不膩，sauce 裡的橄欖油及酒醋的調和比例也很均勻。 蟹肉鑲蘑菇 另一道前菜為蟹肉鑲蘑菇，如球狀的外表，先蟹肉裹上包著香料及胡椒粒的粉後，連著蘑菇下鍋炸後起鍋，咬下時可以嚐到蟹肉的甜美及蘑菇的鮮嫩，是道結合海鮮及蔬菜的料理。 蕃茄濃湯 今天的濃湯是蕃茄濃湯，上桌時香氣立刻散了出來，橙黃色的湯色用湯匙攪時即可感受到湯的濃稠，油炸過的麵包丁大半浮在濃湯	餐前酒 氣泡香檳酒 氣泡香檳酒 生煎日式鮪魚片 生煎日式鮪魚片 生煎鮪魚 醬汁 薑片 蘿蔔絲 鮪魚 胡椒粒 醬汁 鮪魚 牛肉 芥末 芥末 鮪魚 凱散沙拉 羅蔓生菜 起士粉 起士片 麵包丁 橄欖油 酒醋 蟹肉鑲蘑菇 前菜 蟹肉鑲蘑菇 蟹肉 香料 胡椒粒 蘑菇 蟹肉 蘑菇 海鮮

	上，喝了一口，唔，好喝。一整個是濃郁的感覺，讓人忍不住想獨享蕃茄的香氣、微酸的口感整個含在湯裡，果然是道蕃茄「濃」湯 現烤長條麵包 Cream Cheese 現烤的外皮酥脆的長條麵包，單吃的話是單薄了些，不過若蘸桌上的 Cream Cheese，滲入了麵包裡層的 sauce，現烤麵包的軟嫩，外皮的酥脆。滋味立刻大不同的好吃 用完這些後，服務生過來詢問後，就把桌面整理乾淨， 準備上主菜了，美式餐廳上菜速度是頗快的，當然餐點種類沒法餐多也是原因，服務生整理桌面時，還會拿刮尺將桌巾上掉了許多麵包屑的地方，一一仔細慢慢的刮乾淨，十分的貼心。 上主菜時也一併將點的附菜上桌了，所以原本清空的桌面又一下子被佔滿了，共有焗青花椰、馬鈴薯泥、香炒蘑菇及新鮮蘆筍配荷蘭醬 焗青花椰 馬鈴薯泥 香炒蘑菇 新鮮蘆筍配荷蘭醬 荷蘭醬 焗青花椰，焗烤的起士整個包覆住裡面的青花椰，嚐起來不膩，香氣及口感都極佳。 香炒蘑菇是我覺得最好吃的一道小菜，浸過 sauce 的小蘑菇，蒸過出來後再灑上些香料，嚐起來蘑菇細嫩不老，湯汁醇厚。 馬鈴薯泥則是出乎意料的好，用湯匙挖起一角時，湊近一聞可以聞到馬鈴薯蒸過的味道，嚐起來十分細緻綿密而不會整個黏在嘴裡，非常的好吃，蘸著肉汁吃更對味。不過因為易飽足， 所以也是嚐了幾口就停了。 新鮮蘆筍配荷蘭醬是典型的西餐菜，新鮮整根的蘆筍，蒸過後，大把的置於白瓷盤裡，旁邊再放一小碟以蛋黃、檸檬汁、芥末醬、荷蘭芹及奶油打出來的荷蘭醬。食用時可以先咬一口蘆筍，嚐嚐鮮度，再蘸些荷蘭醬，入口有濃濃的奶油香。在吃完大份量的主菜後再回頭吃，可以解膩又健康。	蔬菜 蕃茄濃湯 濃湯 蕃茄濃湯 麵包丁 濃湯 蕃茄 湯 蕃茄 湯 現烤長條麵包 長條麵包 現烤麵包 外皮 主菜 麵包屑 主菜 附菜 焗青花椰 馬鈴薯泥 香炒蘑菇 新鮮蘆筍 荷蘭醬 焗青花椰 馬鈴薯泥 香炒蘑菇 新鮮蘆筍配荷蘭醬 荷蘭醬 焗青花椰 起士 青花椰 香炒蘑菇 小菜 小蘑菇 香料 蘑菇 湯汁 馬鈴薯泥

		馬鈴薯 肉汁 新鮮蘆筍 荷蘭醬 西餐菜 蘆筍 蛋黃 檸檬汁 芥末醬 荷蘭芹 奶油 荷蘭醬 蘆筍 荷蘭醬 主菜
2	期待的進入大門、與服務生說明訂位並跟著到了今晚的位置，我便開始發現了人數實在很多的問題，心理忐忑著，習慣性的走了一圈，然後期望掉了一半。 先來說說食物吧。 烤牛肉、嫩烤霜降沙朗、鮮蔬炒蚌 在燒烤區，有著上面兩項牛肉，本身不吃豬肉、因此一些跟豬有關的食材都沒有注意；而自己拿的這兩個料理，烤牛肉看似鮮嫩多汁，但是卻口感微韌，沒有外觀那樣美味的感覺，淋醬美名為松露黑胡椒醬，其實也只是普通的黑胡椒醬汁，並沒有加入香氣濃郁的松露。而霜降沙朗這個名字，會讓人覺得肉汁豐富、軟嫩可口，但是很抱歉，我沒有吃完這兩塊，這對於甚麼都秉持著要吃完的自己是一種對食物的否決，非常堅韌，是完全不能咬下的口感，令人失望。 而炒蚌肉香氣濃郁，與蔬菜搭配得當，算是佳品。 一來到座位上，變會有兩個小木夾子，讓客人方便點選現炒的菜色，小木夾子如照片所示，粗糙而且有點不衛生，與同等類型的餐廳如上闔屋、饗食天堂變有所區別。 而這盤久待而來鮮蔬炒軟絲，也讓人發覺京采快炒的功力不差，同樣是快炒的印象中還有田雞腿、炒蛤蠣、炒青菜，但是我們並沒有	烤牛肉 嫩烤霜降沙朗 鮮蔬炒蚌 牛肉 豬肉 豬 食材 料理 烤牛肉 淋醬 松露黑胡椒醬 黑胡椒醬汁 松露 霜降沙朗 炒蚌肉 蔬菜 菜色 鮮蔬炒軟絲 田雞腿 炒蛤蠣 炒青菜 熱炒 食材

	點選太多熱炒，而我也逐漸發現每期名為海鮮百匯餐廳，其實並沒有讓很驚艷的食材出現。 生魚片 最惡，也不過如此。 我不知道為甚麼當初爬文時會有版友推薦生魚片，但是我當天差點因為生魚片而找餐廳經理來理論。版友若所研如是，那便是這家餐廳的品管有了問題，總而言之，我非常非常的不滿意，而且我也沒有完全吃完。 照片可以說明的十分清楚：可怕的刀工、可怕的色澤，千萬不要想像它的口感，因為我在寫下這篇文章的當下想起，還是一陣反嘔。不論是鮪魚、旗魚，兩者的口感都是軟爛，而且帶筋，我必須把筋抽出，當成碎肉來吃；而很難雷到的鮭魚，卻一點香氣都沒有，完完全全，一點香氣都沒有！ 生魚片區的生魚片，全部都放置在盤子中，盤子裡面泛著一層水光，很明顯的就是沒有清理的結果，師父一邊切生魚片、一邊啪擦啪擦的跟身邊的人說話、與正妹閒聊，在我看來是十分生氣的，在加上魚生品質實在糟糕，才會讓我失望至極。 也罷，試試其他料理。 揚物 酥炸鮮菇、酥炸花枝、炸銀白魚 銀白魚已經涼掉，鮮菇尚可但卻失去水分，花枝更是口如嚼蠟，同樣出現的是師父的輕薄與怠慢，似乎是害怕客人的食量，總是在一陣閒聊後，才會慢條斯理的開始炸炸蝦，而每次的數量兩手數數剛好，想當然爾，一樣會先顧慮到正妹是否足夠了，雄性動物都餓死吧。 指定菜色 砂鍋鮑魚 拿完揚物回到座位，我們發現座位上有一張小紙條切割成一小段一小段，原來只要拿著夾子夾上交給服務生，就會替你送到座位上。服務生一開始並未提醒這一點，而且繳出紙條到食材送上，那可真是顧客的漫長等待。 這一道的名字我並沒有記下，似乎是砂鍋鮑魚之類的名稱，而這到	生魚片 生魚片 生魚片 鮪魚 旗魚 筋 碎肉 鮭魚 生魚片 生魚片 魚生 揚物 酥炸鮮菇 酥炸花枝 炸銀白魚 銀白魚 鮮菇 花枝 炸蝦 菜色 砂鍋鮑魚 揚物 砂鍋鮑魚 大白菜 料理 鮑魚 象拔蚌 冷盤 沙拉 橙味雁胸 煙燻鮭魚 食材 食材 廉價火鍋 蔬菜 沙拉 沙拉 蔓蘿

	料理我硬將它吞下以避免過份浪費，因為，它已經餿掉了。大白菜的餿味十分明顯，想當然爾，不是今天的料理，鮑魚也不過是象拔蚌，再度失望。 冷盤 自取沙拉、橙味雁胸、煙燻鮭魚 冷盤區可以說是杯盤狼藉，所有的食材都混雜在一起，而食材本身卻會出現廉價火鍋變得蔬菜爛邊，實在令人訝異。勉強的選擇了沙拉，想說沙拉總不會有問題，沒想到蔓蘿竟然老到中間的芯無法吞下，鮭魚更是軟爛、充滿腥味，倒是橙汁雁胸表現不錯，肥嫩的雁肉搭配上酸甜合宜的醬汁，是十分美味的，雖然我懷疑雁肉不過是鴨胸...	芯 鮭魚 橙汁雁胸 雁肉 醬汁 雁肉 鴨胸
3	第一道菜來了，光是外表就讓人覺得很美味， 高腳杯加上架在上面的湯匙，實在吸引人... 內容物看似冰淇淋，經過介紹才知道，原來是馬鈴薯泥佐自製葡萄醬汁，真的好特別，馬鈴薯泥沒有添加調味，沒有黑胡椒、奶油、巴西里...等常用的調味 搭配上酸酸甜甜的葡萄醬，十分適合，還可以吃出馬鈴薯的原味，很不錯的前菜!! 外表很可愛的馬鈴薯沙拉，先吃一口未沾醬汁的馬鈴薯泥，口感綿密細緻，夾雜一點顆粒，很甘甜，配上醬汁一起吃，酸酸甜甜的味道跟馬鈴薯互相融合，讓人胃口大開 新鮮的葡萄醬汁沒有人工香料的味道，酸酸甜甜的滋味，真的好好吃喔～ 第二道菜，枸杞絲瓜，哈，我看得出來啦 絲瓜去皮切塊，稍為煮過，用鹽巴調味，再加上枸杞的甜味，整個把絲瓜的原味都展現出來，沒有過多的調味，像是蛤蜊的鮮味、蝦米、薑絲、蒜頭等 吃起來清脆甘甜，不會煮得過頭而太軟 好吃！ 這道絲瓜完全是品嘗原味，但是絲瓜清脆爽口，而且沒有一般絲瓜的土味，我連湯都喝光光了喔～ 第三道菜，梅子豬排，小小一塊，用野薑花葉襯著，吃過那麼多種炸豬排，還沒吃過包梅子的耶，外皮很酥脆，加上豬肉跟酸酸甜甜梅子，感覺很清爽，梅子消除了油膩感，搭配起來很棒！	菜 冰淇淋 馬鈴薯泥 葡萄醬汁 馬鈴薯泥 黑胡椒 奶油 巴西里 葡萄醬 馬鈴薯 前菜 馬鈴薯沙拉 醬汁 馬鈴薯泥 醬汁 馬鈴薯 葡萄醬汁 人工香料 枸杞絲瓜 絲瓜 鹽巴 枸杞 絲瓜 蛤蜊 蝦米 薑絲 蒜頭

	後來聽老闆娘說他們店是嘉義裡起司豬排的創始店呢！ (嗚嗚 剛剛網誌打到一半，按了一下 backspace，結果內容都不見了，害我重打一次.... ╰(｀皿´＊)╯) 擺盤光看到橫切面就快流口水了，一口咬下，薄薄外皮酥脆但不油膩，梅子酸甜中和配上熟度剛好的豬肉，一口咬下真是幸福 雖然是很小的一道菜，但是已經可以看出廚師的功力了！！ 第四道菜，美味的插曲，因為大聲讚揚好吃而得到的招待菜，奶油炒蛋用漂亮又昂貴的湯匙裝著，僅僅一口的份量，光看起來就很好吃！吃第一口，蓬鬆的蛋、蛋香、奶香、還有甜味，好像在吃鬆軟的蛋捲一般，再搭配上海苔粉，就讓我想到的煎餅... 好吃，但是有點甜，如果吃多了恐怕會膩，就跟我吃蛋糕一樣， 但是他份量抓得剛剛好！不多也不少，厲害！ (喔喔，打完一道菜要存一次檔...避免悲劇再度發生 o(一へ一+)o) 我比較想叫它”蛋糕炒蛋” 口感很像蛋糕，帶著一股奶香味，香甜可口，蛋的熟度剛剛好，吃起來有像蛋糕一樣的蓬鬆感，又很滑嫩，吃到這道特別招待真是開心：) 第五道菜，干貝滷肉飯， 老闆娘說這是台南的作法，肉塊、香菇、干貝、蝦米， 味道濃郁甘純，十分好吃。 (但是我覺得我也可以做出來 XD) 吃不夠還可以加飯喔！ 跟一般外面賣的滷肉飯相較之下，雖然香味沒有那麼濃，但是相較之下沒有油膩膩的肥肉 比較健康，吃起來也不油不膩，帶著蝦米跟干貝的香味，讓平常都只吃半碗飯的我把整碗吃光了 第六道菜是清蒸烏鯧，清蒸魚最能吃出魚的鮮味了 老闆娘說這是水庫裡養殖的淡水魚喔！想必一定是很大一尾吧！ 清蒸魚，搭配上醬油、薑絲和蔥絲，還有很多香油， 想必應該是魚蒸熟了，再使用油淋的方式來淋蔥絲薑絲， (不過如果是我做，我應該還會再撒一些黑胡椒粒；	絲瓜 絲瓜 湯 菜 梅子豬排 野薑花葉 炸豬排 梅子 豬肉 梅子 梅子 起司豬排 梅子 豬肉 菜 菜 招待菜 奶油炒蛋 蛋 海苔粉 煎餅 蛋糕 菜 蛋糕炒蛋 蛋糕 蛋 蛋糕 干貝滷肉飯 肉塊 香菇 干貝 蝦米 滷肉飯 肥肉 蝦米 干貝 飯 清蒸烏鯧

	而我也會懶得把蔥薑切絲，大概就切蔥段跟薑片吧 XD) 好吃！魚的鮮甜滋味，搭配上辣辣的薑絲蔥絲，還有香味十足的醬汁，更加襯托出魚肉的美味本質呢，而且一點腥味都沒有喔！ 剩下的湯汁也一點都不能浪費，還可以來拿辦飯呢～	清蒸魚 魚 淡水魚 清蒸魚 醬油 薑絲 蔥絲 香油 魚 蔥絲薑絲 黑胡椒粒 蔥薑 蔥 薑片 魚 薑絲蔥絲 醬汁 魚肉 湯汁 飯