國立臺灣大學文學院語言學研究所
碩士論文
Graduate Institute of Linguistics
College of Liberal Arts
National Taiwan University
Master Thesis

批踢踢語料庫之建置與應用
PTT Corpus: Construction and Applications

劉純睿
Tsun-Jui Liu

指導教授：謝舒凱博士
Advisor: Dr. Shu-Kai Hsieh

中華民國 103 年 9 月
September, 2014

致謝

首先，我要感謝當初選我進來的教授們－蘇老師、江老師、呂老師、馮老師。謝謝你們給我這個機會讓我在臺大語言所就讀，我將永遠以臺大語言所為榮。感謝呂老師，雖然我不是您的指導學生，您卻總是不吝嗇給我提供幫助，感激之情，實非三言兩語能夠形容。除此之外，我要感謝我生命中的貴人－謝舒凱老師。我知道我不是一個太有紀律及規劃的學生，想必常常為老師帶來困擾與無奈，但老師卻永遠以鼓勵代替指責，而且總是支持我們想要做的事情。在這之中，我帶給老師很多困擾與無奈，但也因為老師的不厭其煩，我著實學到了不少技能，且這些可能是可作為我爾後在外打拚的工具。除了研究，老師亦在我的生活上提供了許多寶貴意見以及經驗分享，老師對於我們就像是朋友，但同時亦為充滿智慧的長者，我真的很幸運能夠跟著老師做事情，謝謝老師！最後，謝謝一路支持我的朋友們。佳音和漾－你們對我來說就是個溫暖的太陽，我永遠不會忘記我在國外落難時你們給我的安慰與鼓勵；他口和伊馮－謝謝妳們常常聽我說一些既無聊囉嗦又冗長的故事，從妳們那邊我得到了很多的快樂；安伯－雖然常常開妳玩笑，但是對於妳的研究精神與對朋友的慷慨，我其實點滴在心；當然，最重要的，謝謝愛米粒－遇見了妳，人生改變許多，也成長不少。妳的地方總是最熱鬧的地方，是大家的開心果，是笑果的保證，有妳在，永遠不怕冷場。還有謝謝 LOPERS，小弟我很榮幸能夠跟大家共事，謝謝大家。

中文摘要

近年來，語料庫為本與語料庫驅動之研究愈來愈受到關注與重視。在台灣華語中，中央研究院平衡語料庫 (Chen et al., 1996) 以及中文十億詞語料庫 (Huang et al., 2005) 為當今兩個最被廣泛使用的語料庫。然而，這些語料庫並不是完全沒有限制。在語料的部份，這些語料庫大多已經停止更新或尚未更新，也就是說，這些資料庫已經無法完全即時反應當代台灣華語的使用狀況。對於眾多研究者來說，在蒐集新興語料上更產生了一定的程度的難度與不便性。正因如此，本篇論文以 PTT（批踢踢）作為資料來源，試圖建立「批踢踢語料庫」— 一個具有自動蒐集、更新、分析及後處理的動態語料庫。除此之外，該語料庫亦會提供一個友善且便利的網路平台，提供研究者作使用。在批踢踢語料庫中，語料的斷詞是透過 Jseg — 一個利用中央研究院平衡語料為訓練基礎之中文斷詞器 — 所達成。而在詞性標註方面，則是採用 Brill Tagger (Brill, 1992) 所使用的演算法，且以中文句結構樹資料庫 (Chen et al., 1999) 中約莫一萬中文句作為訓練的語料。批踢踢語料庫提供了網路介面以供研究者使用，並包含許多根據批踢踢語料所發展出來的應用，其中包括基本的詞語索引器 (Concordancer) 以及搭配詞抽取器 (Collocation extractor)，以及其他諸如表情符號偵測器 (Emoticon detector) 與情緒極性分類器 (Sentiment polarity classifier) 等等之應用。最後，本研究之希望在批踢踢語料庫的建置後，在現代台灣華語中能夠針對新興語料的部分作補充與更新，並且提供實質的語料庫工具，以簡化資料蒐集上的繁瑣及能有系統地分析語料，使得研究者能更加專注在語料本身的分析與發展。

關鍵字：批踢踢、動態語料庫、中文斷詞器、詞性標註、臺灣華語

Abstract

In recent years, corpus-based and corpus-driven studies are getting considerable attentions. In Taiwan Mandarin, two of the most widely used corpora are Academia Sinica Balanced Corpus (Chen et al., 1996) and Chinese Gigawords (Huang et al., 2005). However, both of the corpora have some limitations on the source of the data, and they have not updated for some time, which makes it difficult to collect more recent examples of language uses. Therefore, the aim of this thesis attempts to establish a dynamic corpus, `PTT Corpus`, which can automatically collect, update and process data from PTT (批踢踢), and provide the applications with a user-friendly interface for researchers. Corpora are segmented with Jseg, a Chinese segmentator trained with data from Sinica Corpus, and part-of-speech (POS) tagged by Brill Tagger (Brill, 1992), a POS tagger trained with data trained on the 9999 sentences in the Sinica Treebank (Chen et al., 1999). `PTT Corpus` provides a web interface with several applications, including Concordancer, Collocation extractor, Emoticon Detector, etc. To conclude, establishing `PTT Corpus` may be of importance in enriching the source of modern corpora, providing useful corpus tools, simplifying the analysis of recent language uses and changes in linguists in Taiwan Mandarin.

Key words: PTT, dynamic corpus, Chinese segmentator, POS tagging, Taiwan Mandarin

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research motivation

Recent years have seen growing importance placed on research in corpus-based studies. In Taiwan Mandarin, the most widely used corpus is Academia Sinica Balanced Corpus (Sinica Corpus) (Chen et al., 1996). Sinica Corpus contains ten million words with human segmentation and tagging, which raises the accuracy to over 96%. Another frequently used corpus is Chinese Gigawords (Huang et al., 2005), which contains over 14 billion words with automatic with unsupervised segmentation and tagging. The data can be accessed from Chinese Sketch Engine[1] to perform a variety of tasks such word sketches, syntax analysis thesaurus analysis, etc. However, both of the corpora have some limitations on the source of the data, and they have not updated for some time, which makes it difficult to find more recent examples of language uses. Hence, the goal of this thesis is to develop a dynamic corpus which can automatically collect, update and process data from PTT (批踢踢) in an attempt to capture the emerging uses of language.

[1] http://wordsketch.ling.sinica.edu.tw/

1.2 Significance

PTT Corpus[2], to the best of our knowledge, is the first comprehensive PTT-based corpus system with interactive interface which contains a series of tools for users to explore different aspects of PTT data, such as querying lexical data, making concordance, generating collocations, etc. It complements the lack of modern corpora, which could be a potential source of new words, patterns or constructions. Furthermore, the system is dedicated to provide an user-friendly, open-source and easily accessible interface for the purpose of enhancing the convenience which helps researchers more focus on the research itself. The corpus is still in its infancy, but it is hoped that establishing the system will enrich the source of modern corpora and facilitate the corpus-based linguists in Taiwan Mandarin, and it may simplify the analysis of recent language uses and changes.

1.3 Organization of the thesis

The remainder of this thesis is divided into four chapters. Chapter 2 reviews previous works regarding the issues and examples of the design, establishment and applications of some modern corpora, and research which used PTT as the source or topic will also be addressed. Chapter 3 describes the methodology and procedures for the collection, compilation and preprocessing of the proposed PTT corpora. Chapter 4 describes the applications of PPT Corpus, including adaptive segmentator, emoticon detector, concordancer, collocationer and sentiment polarity classifier. In chapter 5 conclusions are presented and suggestions are made for further research.

[2]http://lopen.linguistics.ntu.edu.tw/PTT

Chapter 2

Literature Review

2.1 Corpus linguistics and the web

After the creation of the World Wide Web (WWW) in 1994, corpus linguistics is confronting with the chance and challenge to use web as an enormous area of linguistic data source, providing abundant accessible resources of human language as a way to realize the multidimensionality of languages. Since the web has become the hot spot of resources for modern corpora, some critical issues must be discussed.

McEnery & Wilson (2001) describes that the modern corpus have the four main characteristics:

1. sampling and representativeness
2. finite size
3. machine-readable form
4. a standard reference

However, some of the issues should be re-considered. The question for *representativeness*, for example, should be changed from "how do you know that your web corpus is representative?" to "how do you know whether any corpus is representative (of what)?" (Kilgarriff, 2001). Another often-mentioned problem of corpus is *balancedness*. Leech (2006) defines *balanced* as the proportionality of the subcorpora and their relative frequency of occurrence of those genres in the language's textual universe as a whole. The author

then stresses some difficulties of how a corpus can be rendered balanced. Manning & Schütze (1999) also suggests that "... having more training data is normally more useful than any concerns of balance, and one should simply use all the text that is available." Lüdeling et al. (2006) also discuss the inherent limitations of relying on a search engine as a data source, and focus on some of the core issues of using the web as corpus, including the source quality and quantity, corpus stability, data annotation, and meta information.

Furthermore, the way of utilizing web as the source of corpora should be classified. In general, following De Schryver (2002), there are two ways of using web as language resource: Web for Corpus (WfC) and Web as Corpus (WaC). Hsieh (2014) provides a clear distinction between WfC and WaC:

> *The former refers to the efforts that aim to "concordance the web" by tailoring search engine to linguistic needs. ... The latter is grounded on a more realistic and technically doable approach by collecting data from the web, and pipelining the working procedures of preprocessing, annotation and indexing, and creating a querying interface for linguistic use.*

Since the amount of data is increasing rapidly and become too much to handle for any human being, the focus is shifting towards the creation of software or interface for the automatic analysis of corpora. Word Sketch, for instance, is one of the first softwares that "automatically-produce[s] summary of a word's behaviour" (Kilgarriff & Tugwell, 2002) Fletcher (2006) also surveys some characteristics of the Web and discusses the pros and cons of exploiting the Web as/for corpus. He reviews several innovative applications of Web data to corpus-related issues, e.g., KWiCFinder (KF), an application to help realize the Web's promise. In sum, the prerequisite for building a proper search engine for linguists is that the process of crawling the web and pre- or post-processing should be automated (Fletcher, 2004; Kilgarriff & Grefenstette, 2003; Volk, 2002)

Many studies have used the web as resources in their research. For instance, Hayashi et al. (1997) developed TITAN, a cross-linguistic search engine for web which provides an interface for users to query in their native language while performing multilingual searches on the web. Its primary goal is to develop an advanced method for indexing the documents

from the web. Renouf (2003) develops `WebCorp`[1], a linguist-friendly tool designed to search the Internet and provide on-line tailored access. The advantages using Web as a corpus are that it is freely-available, covering a range of domains and constantly added to and updated. Fairon et al. (2008) establishes `GlossaNet 2`, a free online concordance service which enables users to search into dynamic Web corpora. Users can define a corpus by selecting RSS[2] feeds in a preselected pool of sources that are visited on a routine by a crawler to generate a dynamic corpus. Users can store their search queries and they will be re-applied on the corpus every time it is updated. The dynamic mechanism allows users to receive the up-to-date concordances with the same queries. Liu & Curran (2006) establish a Web Corpus by collecting web pages to create a topic-diverse collection of 10 billion English words. The results show that, compared to normal search engines, the Web Corpus has better performance on context-sensitive spelling correction and thesaurus extraction. The Edinburgh Twitter Corpus (Petrovic et al., 2010) presented a corpus of almost 100 million tweets which is available for public use. McCreadie et al. (2012) develop a tweet crawling software and detail a methodology for building and distributing Twitter corpora. Finally, they analyze whether the distribution approach remains robust over time.

2.2 Linguistic studies and social media

Social media is defined as "a group of Internet-based applications that build on the ideological and technological foundations of Web 2.0, and that allow the creation and exchange of user-generated content." (Kaplan & Haenlein, 2010).

Su (2003) investigates creative uses of writing systems on PTT. The author discussed and identified four popular creative uses of writing, which are enabled by the written nature of the Internet, the orthographic systems available in the society, and the multilingual situation in Taiwan. These result in a unique mode of communication in its own right. Liu (2012) analyzes the cultural phenomenon of *buzzwords* by manually collecting data from

[1] http://www.webcorp.org.uk

[2] RSS (Rich Site Summary) is a computer file format for delivering regularly changing web content. Many news-related web sites, weblogs and other online publishers syndicate their content as an RSS Feed to their subscribers.

PTT and discusses how they influences the daily life conversation and social interaction.

Liu et al. (2013) attempts to capture two-character neologisms emerged in PTT by adopting the n-gram model with proposed linguistic filter. Evaluation task with 25 subjects was conducted against the system's performance with the calculation of Fleiss' kappa measure. The paper suggests the detection and prediction of neologisms in PTT can be improved and facilitated by observing the features.

Sentiment analysis of text aims to determine the attitude or emotion of a speaker with respect to the language use, and it has gained popularity in many research filed, including linguistics. For example, Shivhare & Khethawat (2012) discuss the emotion recognition based on textual data and reviews techniques often used in emotion detection. Their proposed framework considers the ontology of emotion words and calculates the weight for particular emotion and its counter emotion and then compares the both scores. Chen et al. (2010) construct a dynamic Plurk corpus and conduct experiments by way of combining textual statistics and emoticons data to achieve automatic mood classification of microblogging messages from Plurk. Lu et al. (2013) follows Principle Component Analysis and collocation approach to investigate the relationships within emotion words under linguistic aspect in processing emotion detection.

Chapter 3

Compiling the PTT Corpus

3.1 Introduction of PTT

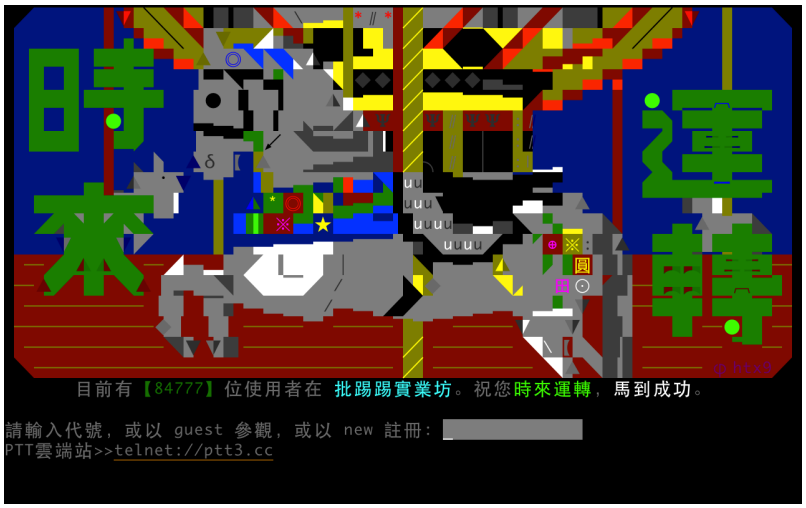


Figure 3.1: PTT login screen

PTT (批踢踢) is a terminal-based bulletin board system (BBS) based in Taiwan, which contains over 20,000 discussion boards with more than 1.5 million registered users, and over 10,000 articles are posted everyday. Figure 3.1 shows the login screen of PTT.

According to Alexa Internet[1], a prestigious company which provides traffic data and global rankings for 30 million websites, PTT is ranked as the top 38th websites in Taiwan, and it has been widely quoted by the public media. The interaction between PTT and

[1]http://www.alexa.com/

public media can be illustrated with the word "服貿" (Service Trade Agreement, a recent public issue discussed intensively in Taiwan) as a keyword to search the post number in PTT Corpus and two large news media in Taiwan, i.e., United Daily News (UDN)[2], The Liberty Times (TLT)[3] as an example. As can be seen in Figure 3.2, PTT has a close relationship with the mainstream media. That is to say, issues discussed in PTT have a higher chance to raise media attention, leading to greater public awareness.

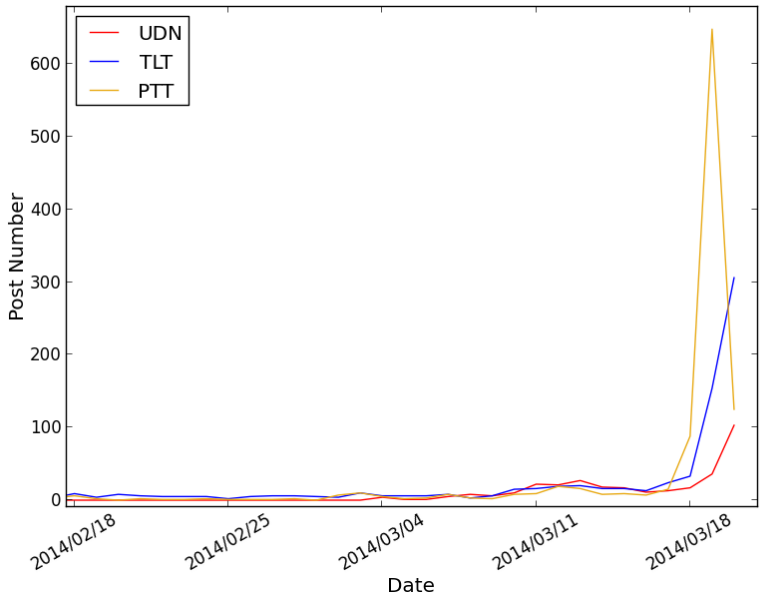


Figure 3.2: Interaction between PTT and public media: 服貿 as an example

In addition, neologisms sometimes could be rapidly distributed in PTT but not yet in public media. The novel word 美江, for example, referring to a Taiwanese priest who delivers several controversial speeches in recent days, can be used as another example to show the interaction of PTT Corpus and other social media, as can be shown in Figure 3.3.

In addition to language use along the temporal dimension, PTT also provides the window into the demographic characteristics of its users.

In PTT Corpus, IP addresses are converted into longitude and latitude and plotted on the map, as can be seen in Figure 3.4 and Figure 3.5, in which the darker the color, the more users there are in the area. It should be also noted that the converted longitude and

[2]http://udndata.com/
[3]http://www.libertytimes.com.tw/

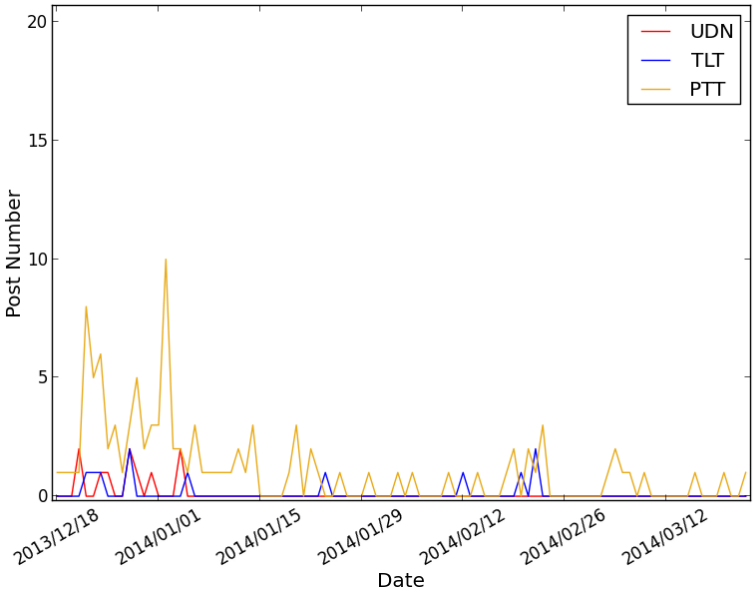


Figure 3.3: Interaction between PTT and public media: 美江 as an example

latitude are estimated location only. In Figure 3.4, it can be observed that the majority of PTT users are from Taiwan, but there are some in other countries as well.

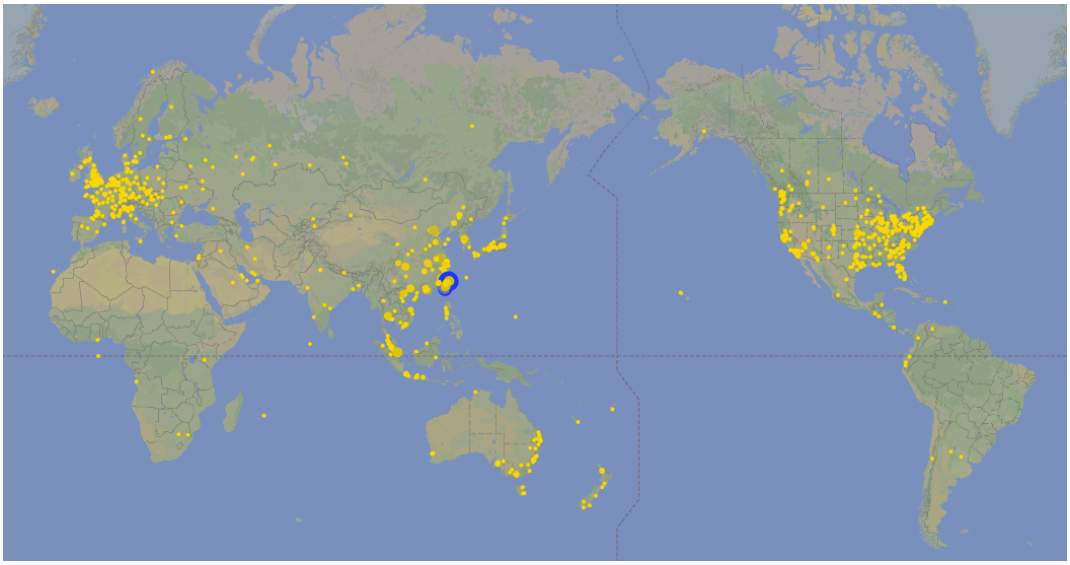


Figure 3.4: PTT user distribution map: world

3.2 Metainformation

In Figure 3.5, we can find that most PTT users are from major cities in Taiwan, e.g., Taipei, Taichung, Kaohsiung, etc.

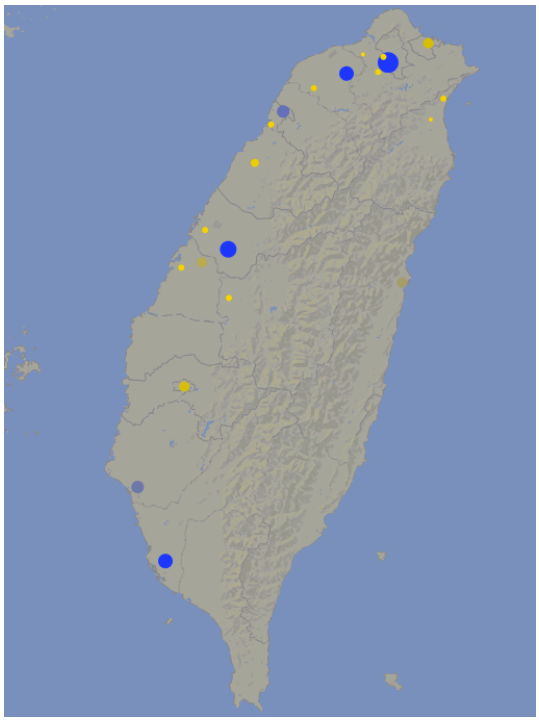


Figure 3.5: PTT user distribution map: Taiwan

Table 3.1 shows the number of post in counties of Taiwan. It should be noted that these numbers are just for the demonstrative purpose in that the these data are collected from only 37 boards. In general, the major posts (users) are from the norther part of Taiwan. This may imply that the language style used in the major city has greater influence on the language development in Taiwanese Mandarin.

Table 3.1: Post numbers of each county

Keelung	158	Changhua	301
Taipei	434,768	Chiayi	164
Taoyuan	1358	Tainan	626
Hsinchu	683	Kaohsiung	1101
Miaoli	65	Hualien	378
Taichung	6592		

3.3 Multidimensionality of PTT data

3.3.1 Language data structure

The basic linguistic structure of a `Post` (Po 文) is comprised of a `heading`, an `article` and `comments`. A `heading` contains a `username` (作者), `title` (標題), `posting time` (時間) and `board name` (看板), which can be seen in Figure 3.6

Figure 3.6: A `heading` of a `Post`

An `article` is the main part of a `Post`, where users can post issues related to the topic of a `board`.

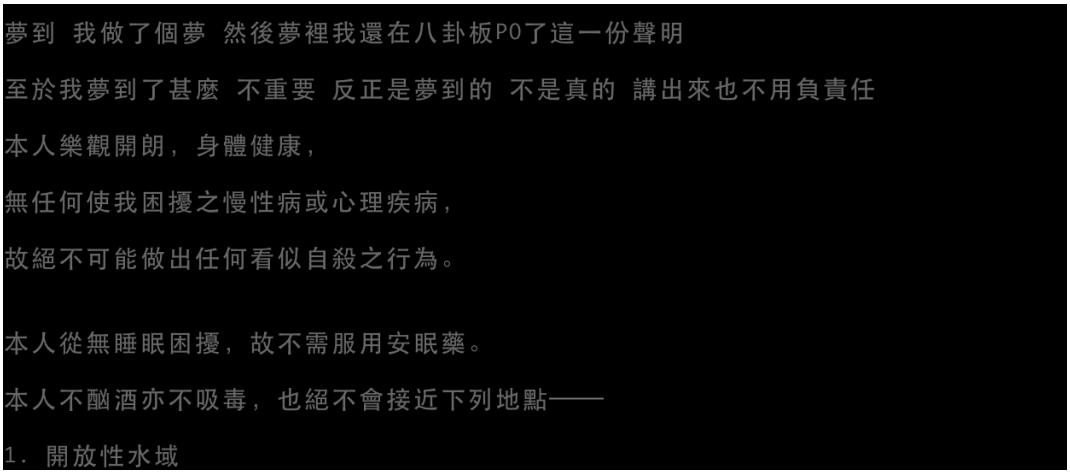


Figure 3.7: An `article` of a `Post`

The last part of a `Post` is `comments`. PTT have a unique commenting system which contains three types of `comment`: `Push` (推), `Boo` (噓) and `Arrow` (→). In PTT, A comment is a sentence with less than 52 half-width characters[4]. Any registered users can show their opinions by choosing one of these types as their attitude toward the posting.The three types of comment are described as the following:

Push (推) Showing supportive, positive attitude for the posting

[4]Since Chinese characters often classified into full-width, it means that a comment can only contains at most 26 Chinese characters.

Boo (噓) Showing disagreement or having opposite opinion of the posting

Arrow (→) Giving comments without showing positive or negative attitude; authors can only give `arrow` to their own postings.

Figure 3.8: An example of comments of PTT

Compared to other social networking service (e.g., Facebook and Twitter), PTT allows users not only to *like* but to *dislike* a post. This provides valuable information for linguistics, sentiment and opinion mining and other research fields.

3.3.2 Data collection

`PTT Corpus` has collected 37 boards so far (details are shown in Table 3.2). Data are collected from 2001 to 2014. The database contains 18,000 active users (users who has posted articles on PTT) with over 70,000 posts and 14 million comments. It is interesting that the number of `Push` is almost six times greater than that of `Boo`, which shows that users tend to show positive attitude toward postings. More details can be seen in Table 3.3.

3.4 Word segmentation

After the data collection, basic pre-processing tasks in compiling corpus in Chinese usually involve two setps: word segmentation and pos tagging.

The most commonly-used Chinese word segmentator is CKIP (Chinese Knowledge and Information Processing) Segmentator[5]. However, there are some drawbacks of the segmentator, as shown in the following:

[5]http://sunlight.iis.sinica.edu.tw/uwextract/demo.htm

Table 3.2: Boards collected in PTT Corpus

Category	Boards
Popular	Gossiping (八卦版) FuMouDiscuss (服貿版) Hate (黑特版) joke (就可版) happy (快樂版) WomenTalk (女人聊天版) StupidClown (笨版) Boy-Girl (男女版) movie (電影版) iPhone (哀鳳版) Beauty (表特版) AllTogether (歐兔版) NBA (NBA 版) Food (food 版)
Northern Taiwan	Taoyuan (桃園版) Keelung (基隆版) ChungLi (中壢版) Hsinchu (新竹版)
Central Taiwan	FengYuan (豐原版) TaichungBun (台中版) TaichungCont (台中市版) Nantou (南投版) Miaoli (苗栗版) Yunlin (雲林版) ChangHua (彰化版)
Southern Taiwan	Tainan (台南版) PingTung (屏東版) Daliao (大寮版) FongShan (鳳山版) Kaohsiung (高雄版) Linyuan (林園版) Chiayi (嘉義版)
Eastern Taiwan	Taitung (台東版) I-Lan (宜蘭版) Hualien (花蓮版)
Islands	Jinmen (金門版) Matsu (馬祖版) PH-sea (澎湖版)

1. The encoding of input strings can only be in Big5[6].

   Words which cannot be encoded in utf-8 will not be processed correctly. For example, 瑠公圳 and 游錫堃 are segmented as follows:

```
游錫(Nb) &#(FW) ２２５３１(Neu) ;(SEMICOLONCATEGORY)
-------------------------------------------------------
&#(FW) ２９７９２(Neu) ;(SEMICOLONCATEGORY)
-------------------------------------------------------
公圳(Na)
```

2. Texts cannot contains xml-like tags.

   When a input string contains angle brackets with letters inside, e.g., <*title*>, the result will not be processed correctly. For example, the following sentence

```
<title>5 伊朗學生溺水 衝浪台生救人</title>
```

   is segmented as:

```
(FW)
-------------------------------------------------------
```

[6]Big5 is a Chinese character encoding method for Traditional Chinese characters.

Table 3.3: Statistics of PTT Corpus (April 14, 2014)

	Number
Board	37
Post	783,990
Users	185,713
Comments	14,330,768
Push	7,363,486
Boo	1,246,135
Arrow	5,721,147
Raw token	176,680,129
Raw type	10,172
Word token	93,945,250
Word type	1,337,117

3. Line breaks relies on some punctuations, such as commas and periods. However, some of the articles on the Web does not use punctuations as the sign of the final of a sentence. For example, the following text

   最近突然想寫看看
   不知道能寫什麼
   有文青要寫什麼的八卦嗎
   肥宅能當文青嗎

   will be treated as:

   最近(Nd) 突然(D) 想(VE) 寫(VC) 看看(Di) 不(D) 知道(VK) 能(D) 寫(VC) 什麼(Nep) 有(V_2) 文青(Nb) 要(D) 寫(VC) 什麼(Nep) 的(DE) 八卦(Na) 嗎(T) 肥宅(Na) 能(D) 當(P) 文青(Nb) 嗎(T)

   which is not appropriate when analyzing texts.

4. Emoticons are treated as a sequence of meaningless of punctuations or characters. For example, ◢[redacted] 崩 ㄟ(〒皿〒)ㄏ 潰 [redacted]◣ will be treated as:

   [redacted](FW) 崩(VH) ㄟ(FW) ((PARENTHESISCATEGORY) 〒(FW) 皿(b) 〒(FW))(PARENTHESISCATEGORY) ㄏ(FW) 潰(VH) [redacted](FW)

   More details for emoticons will be discussed in 4.4.

Therefore, another approach must be adopted in this resaerch. `Jseg`[7], an open-source Chinese Segmentator modified from `jieba` (Sun, 2012) was utilized to segment PTT data. The following will discuss the method and its evaluation.

3.4.1 Method

The segmentation is carried out by the interaction between the lexicon and Hidden Markov Model (HMM). The lexicon is compsed of approximately 2.3 million words extracted from Sinica Corpus. The mechanism is based on the Trie structure to achieve efficient word graph scanning. Sentences using Chinese characters constitute a directed acyclic graph, in which it calculates the maximum probability path based on word frequency combination. That is, for each sentence, the segmentator will assign the frequency to each word if it appears in the lexicon, and then the maximum probability route will be calculated. For out of vocabulary words, HMM was used to classify each sentence as a sequence of B (Begin) I (Intermediate) E (End) S (Single) (as described in Lin & Chang (2006)) and then Viterbi algorithm was used to calculate the optimal probabilty according to the *initial probability*, *transmission probability* and *emission probability* that are trained with Sinica corpus.

Calculatiing probabilities

- Initial probability: The probability of a word beginning with state X in a data set.

```
{”B”: 0.7003435352904435
 ”I”: 0,
 ”E”: 0,
 ”S”: 0.29965646470955654}
```

  Since a word can only starts with B or S, I and E get the probability of zero.

- Transmission probability: The probability for one state to another.

```
{”B”: {”I”: 0.4, ”E”: 0.6},
 ”I”: {”I”: 0.2835512540013088, ”E”: 0.7164487459986912},
```

[7]https://github.com/amigcamel/Jseg

```
"E": {"S": 0.44551469488355755, "B": 0.5544853051164425},
"S": {"S": 0.5138298266610544, "B": 0.48617017333894563}}
```

The above shows that the probability of I and E followed by B, for example, are 0.4 and 0.6 respectively.

- emission probability: The probablity of a character occruing in state X.

For comprehensive explanations with the source code, see Appendix D

3.4.2 Evaluation

In this section, the precision, recall and F1-score of four conditions will be calculated, which are shown as follows:

Condition 1: `jieba` without default dictionary

Condition 2: `jieba` with default dictionary

Condition 3: `Jseg` without ASBC dictionary

Condition 4: `Jseg` with ASBC dictionary

The training data used for evaluation is Sinica Corpus with 19,247 sentences, 11,241,114 tokens and 233,587 types. For each condition, the result of segmentation was compared with the training data, which can be seen in Table3.4

Table 3.4: Evaluation of segmentation results

	Precision	**Recall**	**F1-score**
Condition 1	0.712	0.686	0.698
Condition 2	0.771	0.739	0.755
Condition 3	0.807	0.809	0.808
Condition 4	0.936	0.887	0.911

The result shows that `Jseg` has the F1-score at 0.911 with precision at 0.936 and recall at 0.887, which outperforms `jieba`.

3.5 POS tagging

3.5.1 Method

The part-of-speech (POS) tagger uses Brill Tagger (Brill, 1992) which comes with Python NLTK package (Bird et al., 2009). It is composed of a sequential Backoff chain of TrigramTagger, BigramTagger, UnigramTagger, a -3 AffixTagger, and a DefaultTagger with a default tag of "-NONE-". Data were trained on the 9999 sentences in the Sinica Treebank (Chen et al., 1999) with the NLTK-trainer (Perkins, 2011).

3.5.2 Evaluation

NLTK-trainer tags every sentence of the corpus and counts the frequency of each tag it produces. The trained tagger is 0.98% accurate on Sinica Corpus.

Details are shown in Appendix B. In Appendix B, *Found* shows the number of occurrences of each tag produced by the the tagger, *Actual* shows the actual number of occurrences in the corpus, and Precision and Recall show the performance for each tag. If the Precision is less than one, that means the tagger gave the wrong tag to a word, If the Recall is less than one, it means the tagger gave "-None-" to a word (i.e., a word did not get a tag from the tagger).

Chapter 4

Applications

For the easy accessibility of the `PTT Corpus`, a web interface was created. The applications will be discussed in the following sections.

4.1 Jseg and Segcom

Similar to CKIP Segmentator, `PTT Corpus` also provides a web interface for `Jseg` (details are described in 3.4) for users to segment texts online, as shown in Figure 4.1.

Figure 4.1: A web interface for `Jseg`

The result of *強者我朋友是住在天龍國的魯蛇*, for example, can be seen in Figure

4.2.

Figure 4.2: An example of `Jseg`

`Segcom` is used for comparing the results between CKIP Segmentator and Jseg. Results are shown in parallel with different parts highlighted, as shown in Figure 4.3.

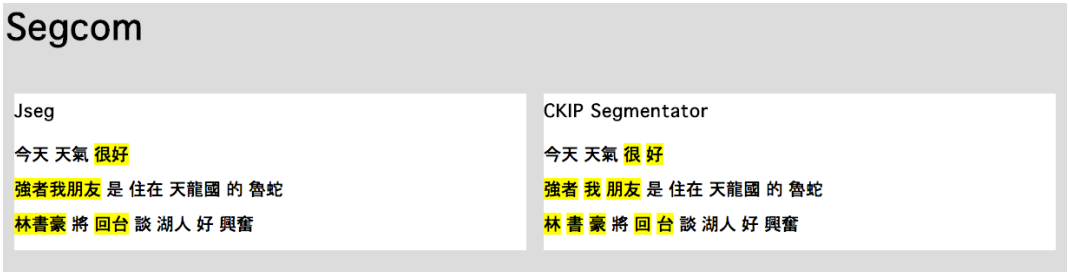


Figure 4.3: An example of `Segcom`

Programmatic use

For those who are familiar with Python[1], here a programmatic use of `Jseg` is provided. This is especially convenient when one needs to process a great amount of data. Notice that `Jseg` has only been tested on Unix-like platform, which may or may not be supported in Windows platform.

One can first retrieve the repository of `Jseg` by using:

```
git clone https://github.com/amigcamel/Jseg
```

Open any preferable Python shell in the terminal, then import `Jseg` as a module.

```
>>> from Jseg import jieba
```

Take the sentence *強者我朋友是住在天龍國的魯蛇* as an example.

```
>>> result = jieba.seg('強者我朋友是住在天龍國的魯蛇')
```

[1] https://www.python.org/

To retrieve the result with POS tags:

```
>>> print result.text()
強者我朋友/Nab 是/V_11 住在/VC1 天龍國/Nca 的/DE 魯蛇/Nab
```

To retrieve the result without POS tags:

```
>>> print result.nopos()
強者我朋友 是 住在 天龍國 的 魯蛇
```

4.2 Concordancer

`PTT Corpus` provides two ways to make concordance for different purposes. One can either search in raw data directly or search with CWB/CQP, which will be described in 4.2.1 and 4.2.2 respectively. The result of the former one contains more "noises", but more comprehensive. On the contrary, the latter one has lesser noises, but the result are less comprehensive since it might be influenced by the accuracy of the segmentation. For the searching speed, searching with CQP/CWB is far more faster than searching in raw data in that data was indexed in advance. The comparison of the two ways of making concordance is simplified as Table 4.1.

Table 4.1: Comparison of making concordance with two methods

	Search in raw data	Search with CWB/CQP
With segmentation	No	Yes
With POS tagging	No	Yes
Indexing	No	Yes
Searching speed	Slow	Fast
Searching results	more comprehensive	less comprehensive
Noises	more	less

4.2.1 Searching in raw data

Figure 4.4 shows the interface of Search Text, in which users can make concordance by querying keywords in either `Post` or `Comments`. One can also use regular expression as the keyword for searching certain sentence constructions, emoticons, numbers, etc. Users

can specify the time span and the width of concordance as well. Boards are currently divided into two major categories, i.e. popular boards and geographic boards, and each category are comprised of several subsets. Figure 4.5 is an example of the result. Each

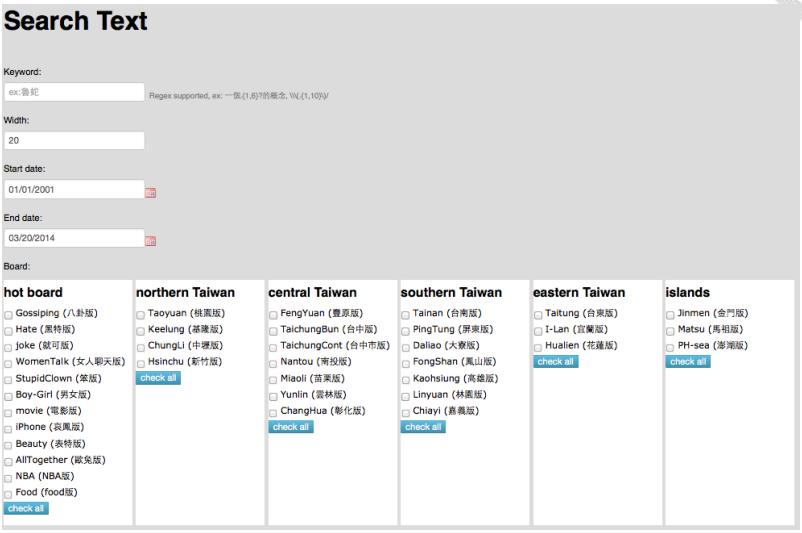


Figure 4.4: Search Text page of `PTT Corpus`

item consists of the source, post time and IP . One can always trace back at the original post by clicking on the selected item.

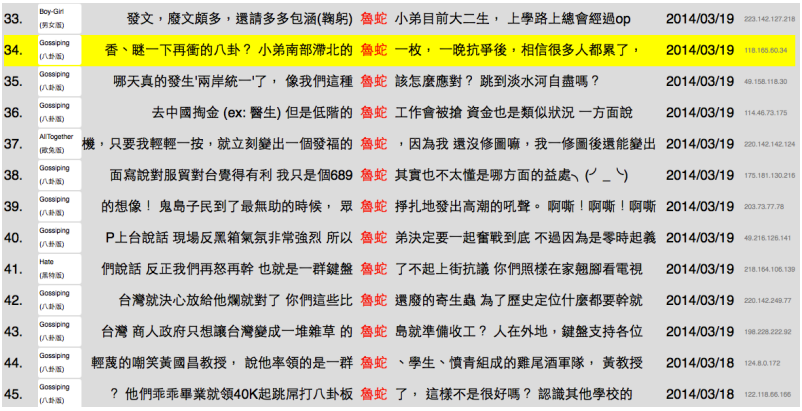


Figure 4.5: Concordance output of *魯蛇*

The input can be a regular expression. For example, one can search emoticons by using the following pattern:

```
\(.{1,10})/
```

and the concordance examples are shown in Table 4.2.

Table 4.2: Concordance examples for emoticons

好不好? 我想要你帶我去日本玩	\(^▽^)/	接著我爸爸就開始很認真的
散發一種吸引人的特質 (對我而言	\(〃ε〃*)/	) 可是可是他大我 14 歲今天早
這樣你就會變成我的啦哈哈 ~~~	\(●′v`●)/	男友: 根本白癡 ~~ 誰說吃下去
.. 阿幹！居然是松鼠欸!!!	\(⊙□⊙)/	這年頭松鼠過馬路都會走天橋 ~
.....(根本沒電) 歐買尬啊啊啊啊	\(Q □ Q)/	!!! 這時女友妹妹突然

Construction can also be found by using the same approach. Liu (2014) uses the following pattern to find some *gei³ ta¹ construction* in `PTT Corpus`, and parts of the results are shown in Table 4.3.

```
給他.{1,4}?下去
```

Table 4.3: Concordance examples for *gei³ ta¹ construction*

!! 第二. 我們是用黑箱作業	給他簽下去	的!! 隱藏麥克風阿鄉親阿
也不用商量找理由，就	給他分手下去	怕對方有因為長期不順和家裡
的資源太慢, 還是有錯字直接	給他開罵下去	!!! 這禮拜是怎樣, 怎麼
是偏夜貓族的但今天不小心 8 點	給他躺床下去	然後不小心做了個劇情離奇的
當然要選個最貴的菲力	給他點下去	。這裡是很正統的那種西餐牛

4.2.2 Searching with CWB/CQP

`PTT Corpus` also implements The IMS Open Corpus Workbench (CWB) (Christ, 1994), a collection of open-source tools for indexing, managing and querying corpora. The central component is a flexible and efficient 'Corpus Query Processor' (CQP) (Christ et al., 1999), a specialized search engine for linguistic research. Evert & Hardie (2011), the recent maintainer of CWB, provides a brief description:

> *The IMS Open Corpus Workbench (CWB) is a powerful and flexible system for indexing and searching corpus data. It is especially designed for use with annotated data, particularly medium to very large corpora with multiple kinds of linguistic annotation at word level (e.g. part-of-speech tags, lemmatisation,*

semantic tags or chunk boundaries). To this end it provides a powerful and flexible data model and query language.

The web interface can be shown in Figure 4.6, which contains three options:

- Corpus: Users can choose to query word in either `Article （文章）` or `Comment （推噓文）`.
- Time order: Users can sorting the results in descending or ascending order.
- POS: Users can decide to show or hide the POS tags in the results. Note that it may take longer to process if `Show` is chosen.

To search 宅男, for example, one can simply query 宅男 (or `[word="宅男"]`). The output is shown in Figure 4.7.

Figure 4.6: Interface of *Concordancer*

Concordance with POS tags will be shown by clicking *`Show POS`* button, as shown in Figure 4.8.

The output of contains of three parts:

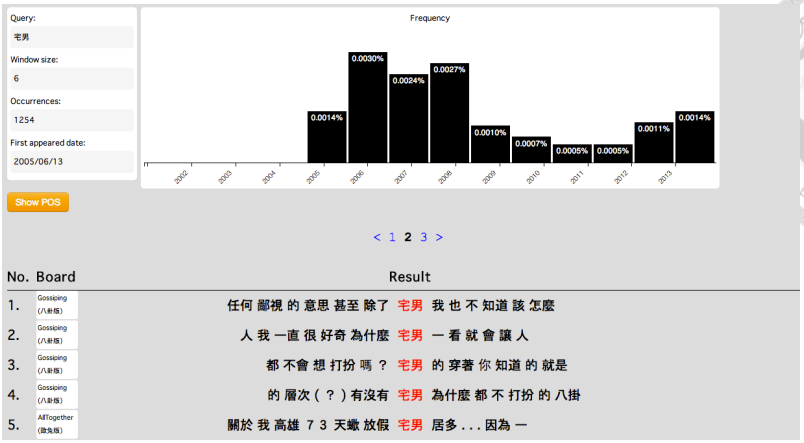


Figure 4.7: Concordance output of 宅男

No.	Board	Result			Post Time
1.	Gossiping (八卦版)	任何/Neqa 鄙視/VJ1 的/DE 意思/Nac 甚至/Dbb 除了/P50	宅男/Nab	我/Nhaa 也/Dbb 不/Dc 知道/VK1 該/Dbaa 怎麼/Dj	2014/01/26
2.	Gossiping (八卦版)	人/Nab 我/Nhaa 一直/Dd 很/Dfa 好奇/VH11 為什麼/Dj	宅男/Nab	一/Neu 看/VC2 就/Dd 會/Dbaa 讓/VL4 人/Nab	2014/01/26
3.	Gossiping (八卦版)	都/Dab 不會/Dbaa 想/VE2 打扮/VC2 嗎/Td ？/PUNCTUATION	宅男/Nab	的/DE 穿著/VC31 你/Nhaa 知道/VK1 的/DE 就是/Cbba	2014/01/26
4.	Gossiping (八卦版)	的/DE 層次/Nac (/PUNCTUATION ？/PUNCTUATION)/PUNCTUATION 有沒有/Dj	宅男/Nab	為什麼/Dj 都/Dab 不/Dc 打扮/VC2 的/DE 八掛/VC33	2014/01/26
5.	AllTogether (歡樂版)	關於/P41 我/Nhaa 高雄/Nca 7 3 /Neu 天蠍/-None-放假/VH11	宅男/Nab	居多/Neqa ./PUNCTUATION ./PUNCTUATION ./PUNCTUATION 因為/Cbaa 一/Neu	2014/01/25
6.	AllTogether (歡樂版)	懶得/DE 出門/Nca 時/Ng 就/Dd 在/P21 家當/P16	宅男/Nab	狂案/Nac 右鍵/Nac 或/Caa 看/VC2 小說/Nac 幻想/Nv1	2014/01/24
7.	Boy-Girl (男女版)	臺灣/Nca 各/Nes 地/Naa ， /PUNCTUATION 我/Nhaa 的/DE	宅男/Nab	個性/Nad 被/P02 她/Nhaa 完全/Dab 顛覆/Nv1 。/PUNCTUATION	2014/01/23

Figure 4.8: Concordance of 宅男 with POS tagging

1. Metainfo: The upper-left box shows the number of occurrences of the query word and the date of its first appearance in `PTT Corpus`. 宅男, for example, occurs 1254 times, and it first appeared at 13 June, 2005.

2. Frequency: The upper-right box is a bar chart showing the the frequency of the query word for each year.

3. Concordance: The major part of the page. Each row is linked to the original post with the query word highlighted, as shown in Figure 4.9

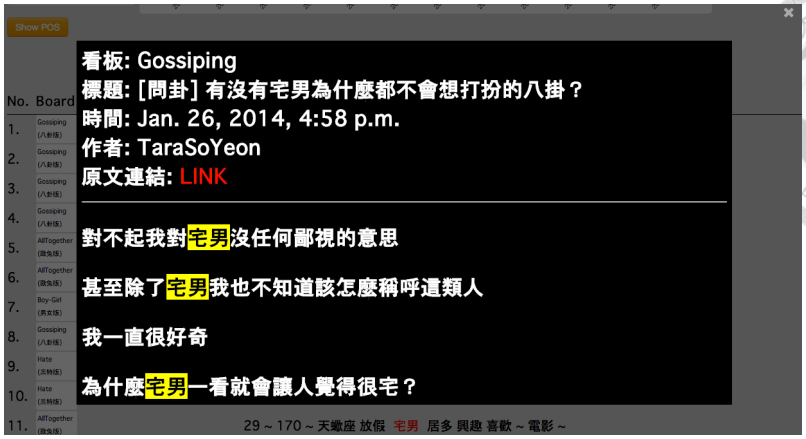


Figure 4.9: Original post with query word highlighted

4.3 Grammatical collocation extractor

A collocation is "the habitual co-occurrence of individual lexical items" (Crystal, 2011). In other words, it is a sequence of words or patterns that co-occur more often than would be expected by chance.

Sketch Engine further utilizes Sketch Grammar, an extended version of CQP language (Evert, 2005) which uses regular expressions over POS-tags for querying corpora, to extract potential collocations. For instance, the following rule

```
[pos='V'] [pos='(DET|NUM|ADJ|ADV)']* [pos='N']
```

is used to retrieve the verb-object relation in English, which extracts the data containing a verb followed by a noun with optional determiners, numerals, adjectives and adverbs preceding the noun, e.g.*send mails*, *sending a registered letter*, *sent the/her/two ordinary letter*, etc.

Similarly, Chinese Sketch Grammar[2], a Chinese version of Sketch Grammar, is adopted and transformed into standard CQP language to extract collocation candidates in `PTT Corpus`. For example, the following rule[3]

```
2:[tag="A"|tag="VH11"|tag="VH13"|tag="VH21"] [word="的"]
   [tag="N[abcd].*" & tag!="Ncd"]{0,2} 1:[tag="N[abcdhf
```

[2]http://wordsketch.ling.sinica.edu.tw/cws/run.cgi/wsdef (Users should apply for an account to view the content.)

[3]Descriptions of POS tags can be seen in Appendix E

```
].*" & tag!="Nbc.*" & tag!="Ncd.*" & word!="者" & word
!="們"] [tag!="N[abcdhef].*"|tag="Nbc.*"|tag="Ncd.*"]
```

which is transformed in standard CQP language as below

```
@TAR:[pos="A"|pos="VH11"|pos="VH13"|pos="VH21"] [word="的
"] [pos="N[abcd].*" & pos!="Ncd"]{0,2} KEY:[pos="N[
abcdhf].*" & pos!="Nbc.*" & pos!="Ncd.*" & word!="者"
& word!="們"] [pos!="N[abcdhef].*"|pos="Nbc.*"|pos="
Ncd.*"] :: KEY & TAR
```

describes the grammatical relation between adjective modifiers and an modifies, which can be simplified as the following sequence:

```
TAR Wx (Wy) KEY Wz
```

Details of each symbols are explained as follows:

- TAR is the collocate whose POS can be *A*, *VH11*, *VH13* or *VH21*.
- KEY is the headword which can be any word but *者* and *們*, and the POS can be *Na**, *Nb**, *Nc**, *Nd**, *Nh** and *Nf** excluding *Nbc** and *Ncd**.
- Wx is specifies as *的*.
- Wy is an optional word with length at most two whose POS can be *Na**[4], *Nb**, *Nc**, *Nd**, *Nf** and *Nh** but not *Ncd*.
- Wz is a word that cannot be *Na**, *Nb**, *Nc**, *Nd**, *Nh**, *Ne** and *Nf** except *Nbc** and *Ncd**.

Chinese Sketch Engine adopts Mutual Information (MI) to measure the salience of a collocation, and the algorithm of MI is

$$MI\text{-}score = \log_2 \frac{fxyN}{fxfy}$$

[4] * means any letter(s)

where fx is the number of occurrences of word X, fy is the number of occurrences of word Y, fxy is the number of co-occurrences of words X and Y, and N is the corpus size, i.e., the number of tokens in the corpus. MI scores provides a measure of the association degree of a given word with others. However, MI score is not without its flaw. As suggested by Kilgarriff & Kosem (2012), "One flaw of the original work is that MI emphasises rare words (and an ad hoc frequency threshold has to be imposed or the list would be dominated by very rare items)." That is, MI score brings up words which are not frequent in the corpus. Thus, logDice (Rychlỳ, 2008) is implemented in `PTT Corpus` as the measurement of collocations.

The algorithm of logDice is

$$logDice = 14 + \log_2 \frac{2fxy}{fx + fy}$$

in which 14 is a theoretical maximum, in case when all occurrences of X co-occur with Y and all occurrences of Y co-occur with X. The value, however, is usually less then 10. `PTT Corpus` used the logDice algorithm adjusted by Sketch Engine, which is

$$logDice = 14 + \log_2 \frac{2 \cdot ||w1, R, w2||}{||w1, R, *|| + ||*, *, w2||}$$

where $||w1, R, w2||$ represents the number of occurrences of the headword ($w1$) and its collocate ($w2$), $||w1, R, *||$ represents number of occurrences of the headword in the grammatical relation (R) with any other collocate ($*$), and $||*, *, w2||$ represents number of occurrences of the collocate in any grammatical relation with any headword. Hence, the equation is based not only on the frequency of the two words, but also on the frequency of the headword in a certain grammatical relation with the collocate in any other grammatical relation ship. Also, in contrast to MI score, the score of logDice is without regard to the corpus size, which implies that extraction of collocations in small corpora is still valid and feasible.

To find collocations of a word in `PTT Corpus`, one can simply query the word by choosing *Collocation* function. Figure 4.10 is the output for 服貿, as an example. The

output shows the grammatical relations of *服貿*, such as possessor and modifies, and each box contains its collocates, frequency of collocates and the score of logDice. For example, *退回* is the object of *服貿*, and they co-occur 20 times in `PTT Corpus` with the logDice score of 11.0365.

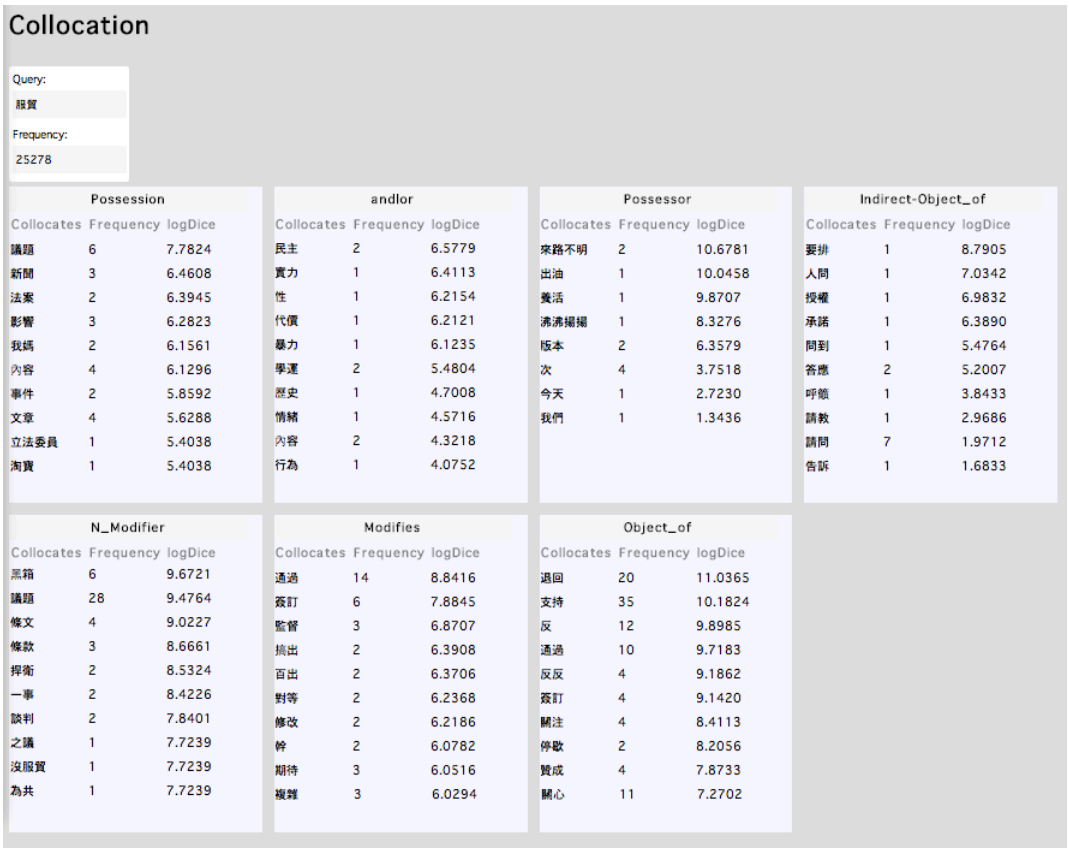


Figure 4.10: Collocates of *服貿*

4.4 Emoticon detector

An emoticon is a combination of characters, letters, numbers or punctuations which is used to express a person's feelings mood or emotion. It has been observed that emoticons are pervasively used on the Web, and PTT is no exception. The importance of emoticons has drawn great attention from linguistics, sentiment analysis, NLP, etc. According to Ptaszynski et al. (2010), emoticons can be divided into three categories as follows, in which `Emoticon detector` will only focus on Type 1 and Type 2.

Type 1: Western type, e.g., XD, :-(, ;).

Type 2: Eastern type (a.k.a *Kaomoji*), e.g., (*°∀°*), ◢▬▬▬ 溫 ╰(〃‿〃)╯ 馨 ▬▬◣.

Type 3: Multi-line ASCII art type, e.g.,

```
ロ卡ロ卡～吃光光
| ⊙  ⊙ |  /
|▼▼▼▼|
|▲▲▲▲|
<▬▬>
 | |
```

PTT Corpus adopts the lexicon-based approach to extract emoticons from PTT. An emoticon lexicon which contains 782 common emoticons was built to detect if a sentence or text contains any emoticons. So far, over 125,000 emoticons has been recognized in PTT Corpus, and the amount is still increasing. Figure 4.11 shows the frequency distribution of emoticons in PTT Corpus, in which x-axis represents the year and y-axis represents the frequency.

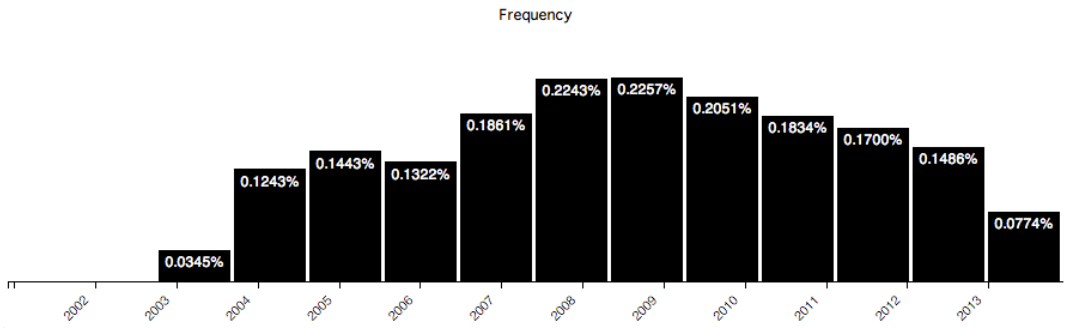


Figure 4.11: Frequency distribution of emoticons in PTT Corpus

The online demo of Emoticon Detector and the result are shown in 4.12 and 4.13 respectively.

4.5 Sentiment polarity classifier

PTT Corpus also implements a rudimentary classifier, Sentipol Classifier, in an attempt to capture the sentiment polarity of texts or sentences from PTT.

Figure 4.12: Interface of `Emoticon Detector`

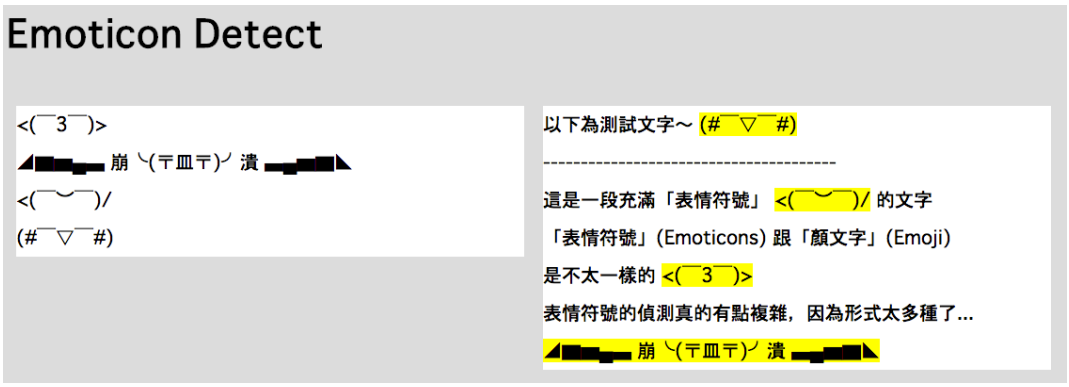


Figure 4.13: An example of `Emoticon Detector`

4.5.1 Data training

`Sentipol Classifier` adopts the unsupervised classification method. First, One third of posts in each board from `PTT Corpus` were retrieved, and each post was segmented with `Jseg` and the polarities are determined by comparing the number of positive words and negative words that occur in each post. The number of positive and negative words were calculated by comparing texts with *mixed emotion-describing word list* (Lu et al., 2013), a word list composed of Standard Stimuli and Normative Responses of Emotions[5] and NTUSD (Ku & Chen, 2007) with 9366 positive words and 11,231 negative words. Next, following Jalilvand & Salim (2012), the number of positive words and negative words are counted in each post, and their numbers are compared. The

[5]http://ssnre.psy.ntu.edu.tw/

polarity score is defined as:

```
Number of positive words - Number of negative words}
```

Finally, 2000 posts with the highest and lowest positive and negative polarity scores are classified as positive and negative respectively.

4.5.2 Evaluation

A ten-fold cross-validation was used for the evaluating three common classifiers: Naive Bayes (NB) classifier, k-nearest neighbors (KNN) classifier and support vector machine (SVM) classifier. Ten tests were performed on each classifier, and each time the classifier partitions the trained feature-weight into different subsets for training and testing. The cross validation can avoid always using the same dataset for training which may generates some biased results. Results are shown in Table 4.4.

Table 4.4: Evaluation of different classifiers

	Accuracy	**Precision**	**Recall**	**F1-score**
NB	0.528	0.460	0.430	0.444
KNN	0.826	0.818	0.758	0.787
SVM	0.877	0.846	0.826	0.836

As can be seen, SVM classifier has the best performance, so `Sentipol` will use SVM classification method.

To enhance the performance of the classifier, positive and negative words are used as features and their frequencies are counted as weight. Features and weights are pairwise, and they are put into `Sentipol classifier`. In sentence *批踢踢真的很方便, 難怪那麼多人喜歡*, for example, the positive words *方便* and *喜歡* are recognized as features, and their frequencies are counted as the weights. 12,409 out of 20,597 words are used as features (details are shown in Appendix A.), in which there are 6008 positive words and 6401 negative words.

A simple calculation of the percentage of posts in each board can be made by using the classifier. In Table 4.5, we use *Gossiping*, *Hate*, *FuMouDiscuss* and *joke* as examples.

Table 4.5: Polarity percentage of posts in *Gossiping*, *Hate*, *FuMouDiscuss* and *joke*

Board	Positive	Negative	Neutral
Gossiping (八卦版)	0.591	0.206	0.198
Hate (黑特版)	0.115	0.792	0.093
FuMouDiscuss (服貿版)	0.564	0.300	0.136
joke (笑話版)	0.340	0.333	0.328

`PTT Corpus` also provides a web interface for sentiment polarity classifier `Sentipol Classifier`, as shown in Figure 4.14.

Figure 4.14: Interface for `Sentipol Classifier`

Figure 4.15 is the output of `Sentipol Classifier`. In the result, Positive words and negative words are highlighed with yellow and blue color, and the the polarity and emotion words that are used by the `Sentipol Classifier` of the post will be shown in the lower boxes.

Figure 4.15: An example of `Sentipol Classifier`

Chapter 5

Conclusions

By means of this study, we have inspected two major topics: the construction of `PTT Corpus` and the applications by exploiting PTT corpora. In chapter 3, we describe the methodology and procedures for the collection, compilation and preprocessing of the proposed PTT corpora. In particular, we have shown the multidimensionality of PTT corpora, in which it contains abundant metainformation with the value of multiple research purposes. Also, we have also discussed the method of word segmentation used in `PTT Corpus`. HMM and DAG with the training data from Sinica Corpus were used in the segmentator with F1-score at 0.91. Furthermore, Brill tagger with training data from Sinica Treebank was used as the POS tagger with the accuracy at 0.98.

In Chapter 4, we introduce five applications of `PPT Corpus`, which are summarized as follows:

- `Segcom`: is used for comparing the results between CKIP Segmentator and Jseg.
- `Concordancer`: an interface for user to make concordance of a query in question. One can query in raw data or query with CWB/CQP.
- `Grammatical collocation extractor`: a corpus tool for extracting collocation by using Chinese sketch grammar and with logDice.
- `Emoticon detector`: a module for detecting emoticons.

- `Sentiment polarity classifier`: an SVM classifier for classifying the sentiment polarity of a given text or sentence.

To conclude, `PTT Corpus`, to the best of our knowledge, is the first comprehensive PTT-based corpus system with interactive interface which contains a series of tools for researchers to explore multidimensionality of PTT data. Also, the system is dedicated to provide an user-friendly, open-source and easily accessible interface. The system provides useful applications to simplify the analysis, helping the researcher more focus on the content of the research. Although `PTT Corpus` is still in its infancy, we hope that the establishment of `PTT Corpus` may contribute in supplementing the source of modern corpora in Taiwan Mandarin, and facilitate the corpus-based/corpus-driven linguists in Taiwan Mandarin.

Appendix A

The weight of selected features

Table A.1: Top 50 features and their frequencies of *Post*

Positive		Negative	
Word	**Frequency**	**Word**	**Frequency**
覺得	0.0114 %	幹	0.0194 %
還是	0.0114 %	不會	0.0081 %
時間	0.0073 %	最後	0.0070 %
已經	0.0072 %	不要	0.0052 %
還有	0.0067 %	問題	0.0045 %
其實	0.0067 %	甚至	0.0036 %
喜歡	0.0059 %	無法	0.0029 %
很多	0.0058 %	幹你娘	0.0028 %
希望	0.0057 %	不能	0.0028 %
感覺	0.0051 %	媽的	0.0027 %
發現	0.0038 %	過去	0.0026 %
非常	0.0037 %	可是	0.0026 %
當然	0.0036 %	他媽的	0.0020 %
應該	0.0035 %	分手	0.0020 %
認為	0.0035 %	離開	0.0019 %

Continued on next page

Table A.1 – *Continued from previous page*

Word	Frequency	Word	Frequency
以為	0.0032 %	如此	0.0019 %
為了	0.0031 %	而已	0.0018 %
不錯	0.0028 %	怎樣	0.0018 %
成為	0.0028 %	竟然	0.0017 %
表示	0.0027 %	不然	0.0016 %
就會	0.0026 %	不用	0.0016 %
發展	0.0026 %	你他媽	0.0016 %
表現	0.0026 %	不管	0.0015 %
想要	0.0025 %	機車	0.0015 %
好像	0.0025 %	還沒	0.0014 %
直接	0.0024 %	不爽	0.0014 %
告訴	0.0024 %	反正	0.0014 %
都會	0.0024 %	都沒	0.0013 %
出現	0.0024 %	不是	0.0013 %
發生	0.0024 %	歷史	0.0013 %
大概	0.0023 %	都沒有	0.0013 %
努力	0.0022 %	也沒有	0.0012 %
繼續	0.0021 %	生氣	0.0012 %
總是	0.0021 %	不太	0.0012 %
第一	0.0020 %	設計	0.0012 %
同時	0.0020 %	沒辦法	0.0012 %
機會	0.0020 %	沒想到	0.0011 %
還要	0.0020 %	可惜	0.0011 %
也許	0.0019 %	不敢	0.0011 %
似乎	0.0019 %	你媽	0.0010 %
在一起	0.0019 %	擔心	0.0010 %

Continued on next page

Table A.1 – *Continued from previous page*

Word	**Frequency**	**Word**	**Frequency**
第一次	0.0018 %	不懂	0.0010 %
根本	0.0018 %	難過	0.0010 %
年輕	0.0018 %	反而	0.0010 %
也會	0.0018 %	他媽	0.0010 %
音樂	0.0018 %	並沒有	0.0010 %
很好	0.0018 %	並不	0.0010 %
愛情	0.0017 %	很難	0.0010 %
自由	0.0017 %	不夠	0.0010 %
實在	0.0017 %	沒什麼	0.0010 %

Appendix B

The Brill Tagger

Table B.1: Coverage metrics

Tag	Found	Actual	Precision	Recall
A	507	506	0.99069767441	0.99532710280
Caa	2914	2914	1.0	1.0
Caa[P1]	24	44	1.0	0.83333333333
Caa[P1]	1	1	1.0	1.0
Caa[P2]	35	41	1.0	0.8
Caa[P2]	1	1	1.0	1.0
Cab	240	239	1.0	1.0
Cbaa	363	373	1.0	0.92592592592
Cbab	5	5	1.0	1.0
Cbba	135	130	1.0	1.0
Cbbb	46	53	1.0	0.5
Cbca	703	700	1.0	0.96551724137
Cbcb	150	158	1.0	0.74074074074
DE	7194	7095	0.83333333333	1.0
DM	2765	2771	1.0	0.99612027158
Daa	263	285	1.0	0.88888888888
Dab	670	670	1.0	1.0
Dbaa	759	838	1.0	0.90909090909
Dbab	1439	1441	0.96	0.92307692307
Dbb	1552	1606	1.0	0.97727272727
Dbc	7	7	1.0	1.0
Dc	662	674	1.0	1.0
Dd	2603	2532	1.0	0.97409326424
Dfa	951	948	1.0	0.97916666666
Dfb	18	20	1.0	1.0
Dg	33	33	1.0	1.0
Dh	473	478	1.0	0.99401197604

Continued on next page

Table B.1 – *Continued from previous page*

Tag	Found	Actual	Precision	Recall
Di	1368	1350	0.85714285714	0.92307692307
Dj	135	134	1.0	1.0
Dk	49	59	1.0	0.94736842105
Head	2	2	1.0	1.0
I	9	9	1.0	1.0
Naa	835	836	1.0	0.99401197604
Nab	9501	9520	1.0	0.99794941900
Nab[+SPO]	3	3	1.0	1.0
Nac	4459	4478	1.0	0.99636032757
Nac[+SPO]	2	2	1.0	1.0
Nad	3885	3873	1.0	0.99707174231
Nad[+SPO]	2	2	1.0	1.0
Naea	201	199	1.0	1.0
Naeb	784	785	1.0	0.99610894941
Nba	2012	2019	1.0	0.99650959860
Nbc	146	149	1.0	0.975
Nca	2456	2454	1.0	1.0
Ncb	2302	2294	1.0	0.99847094801
Ncc	320	319	1.0	1.0
Ncda	1253	1254	1.0	1.0
Ncdb	279	279	1.0	0.98734177215
Nce	110	110	1.0	1.0
Ndaaa	32	32	1.0	1.0
Ndaab	76	76	1.0	1.0
Ndaac	5	5	1.0	1.0
Ndaad	85	85	1.0	1.0
Ndaba	92	93	1.0	0.83333333333
Ndabb	73	73	1.0	1.0
Ndabc	144	145	1.0	0.88888888888
Ndabd	378	378	1.0	1.0
Ndabe	203	199	1.0	1.0
Ndabf	74	74	1.0	1.0
Ndc	32	32	1.0	1.0
Ndda	120	121	1.0	0.92
Nddb	96	98	1.0	0.88888888888
Nddc	282	284	1.0	0.97777777777
Nep	553	542	1.0	1.0
Neqa	895	898	1.0	1.0
Neqb	4	5	1.0	1.0
Nes	342	340	1.0	0.90909090909
Neu	727	707	1.0	0.99333333333
Nfa	17	14	1.0	1.0
Nfc	1	1	1.0	1.0
Nfd	1	2	1.0	0.5
Nfg	19	18	1.0	1.0

Continued on next page

Table B.1 – *Continued from previous page*

Tag	Found	Actual	Precision	Recall
Nfi	1	1	1.0	1.0
Ng	1155	1139	1.0	1.0
Nhaa	2723	2728	1.0	0.95238095238
Nhab	313	306	1.0	1.0
Nhac	64	63	1.0	1.0
Nhb	45	44	1.0	1.0
Nhc	15	16	1.0	1.0
Nv1	869	877	1.0	0.99340659340
Nv2	83	84	1.0	1.0
Nv3	55	56	1.0	1.0
Nv4	871	885	1.0	0.98884758364
P01	1	1	1.0	1.0
P02	210	213	1.0	1.0
P03	146	148	1.0	1.0
P04	58	67	1.0	1.0
P06	121	120	1.0	1.0
P07	338	354	1.0	0.66666666666
P08	1	2	1.0	1.0
P09	2	2	1.0	1.0
P10	2	1	1.0	1.0
P11	251	246	1.0	1.0
P11[+part]	1	2	1.0	0.5
P12	6	6	1.0	1.0
P13	18	16	1.0	1.0
P14	1	4	1.0	1.0
P15	12	12	1.0	1.0
P16	34	34	1.0	0.66666666666
P17	1	1	1.0	1.0
P18	8	8	1.0	1.0
P19	161	154	1.0	1.0
P20	13	14	1.0	1.0
P20[+part]	1	2	1.0	0.5
P21	1232	1190	1.0	0.75
P21[+part]	1	2	1.0	0.5
P22	3	3	1.0	1.0
P23	141	141	1.0	1.0
P24	5	7	1.0	0.66666666666
P25	1	1	1.0	1.0
P26	21	21	1.0	1.0
P27	12	13	1.0	1.0
P28	2	2	1.0	1.0
P29	0	1	None	0.0
P30	21	21	1.0	1.0
P31	312	310	1.0	1.0
P31[+part]	12	12	1.0	1.0

Continued on next page

Table B.1 – *Continued from previous page*

Tag	Found	Actual	Precision	Recall
P32	37	36	1.0	1.0
P32[+part]	1	2	1.0	0.5
P35	207	211	1.0	0.8
P35[+part]	2	2	1.0	1.0
P36	1	1	1.0	1.0
P37	39	39	1.0	1.0
P38	14	14	1.0	1.0
P39	78	79	1.0	1.0
P40	3	3	1.0	1.0
P41	34	32	1.0	1.0
P42	21	24	1.0	1.0
P43	27	27	1.0	1.0
P45	1	1	1.0	1.0
P46	6	6	1.0	1.0
P47	41	38	1.0	1.0
P48	5	5	1.0	1.0
P49	43	43	1.0	1.0
P50	26	26	1.0	1.0
P52	1	3	1.0	0.33333333333
P54	17	17	1.0	1.0
P55	62	63	1.0	1.0
P55[+part]	2	2	1.0	1.0
P58	14	14	1.0	1.0
P59	34	34	1.0	1.0
P60	11	12	1.0	1.0
P61	161	177	1.0	1.0
P62	121	120	1.0	1.0
P63	68	67	0.66666666666	1.0
P64	4	4	1.0	1.0
P66	1	1	1.0	1.0
Str	0	3	None	0.0
Ta	538	662	1.0	1.0
Tb	28	28	1.0	1.0
Tc	229	229	1.0	1.0
Td	66	67	1.0	0.875
VA	2	2	1.0	1.0
VA11	515	511	0.99473684210	1.0
VA11[+ASP]	3	3	1.0	1.0
VA11[+NEG]	1	1	1.0	1.0
VA12	196	199	1.0	0.98360655737
VA13	125	124	1.0	1.0
VA2	47	47	1.0	1.0
VA2[+SPV]	1	1	1.0	1.0
VA3	11	10	1.0	1.0
VA3[+ASP]	1	1	1.0	1.0

Continued on next page

Table B.1 – *Continued from previous page*

Tag	Found	Actual	Precision	Recall
VA4	959	956	1.0	1.0
VA4[+ASP]	18	18	1.0	1.0
VA4[+NEG,+ASP]	1	1	1.0	1.0
VA4[+SPV]	2	2	1.0	0.5
VB11	91	92	1.0	0.98765432098
VB11[+ASP]	1	1	1.0	1.0
VB11[+NEG]	1	1	1.0	1.0
VB11[+SPV]	1	1	1.0	1.0
VB12	67	65	1.0	1.0
VB12[+ASP]	1	1	1.0	1.0
VB12[+NEG]	1	1	1.0	1.0
VB2	20	20	1.0	1.0
VB2[+NEG]	1	1	1.0	1.0
VC1	846	848	1.0	1.0
VC2	3317	3304	1.0	0.99410774410
VC2[+DE]	5	5	1.0	1.0
VC2[+NEG]	13	13	1.0	1.0
VC31	1189	1188	1.0	0.99717514124
VC31[+DE]	1	1	1.0	1.0
VC31[+NEG]	8	8	1.0	1.0
VC31[+SPV]	0	1	None	0.0
VC32	185	180	1.0	1.0
VC32[+SPV]	1	1	1.0	1.0
VC33	401	401	1.0	1.0
VD1	273	264	1.0	1.0
VD2	22	22	1.0	1.0
VD2[+NEG]	1	1	1.0	1.0
VE11	64	64	1.0	1.0
VE12	121	121	1.0	1.0
VE2	1621	1627	0.99183673469	0.99590163934
VE2[+DE]	7	7	1.0	1.0
VE2[+NEG]	8	8	1.0	1.0
VF1	132	130	1.0	0.96969696969
VF2	227	222	1.0	0.96153846153
VG1	196	202	1.0	0.98734177215
VG1[+NEG]	4	4	1.0	1.0
VG2	553	543	1.0	0.99019607843
VH11	3731	3722	1.0	0.99653018737
VH11[+ASP]	7	7	1.0	1.0
VH11[+DE]	1	1	1.0	1.0
VH11[+NEG]	1	1	1.0	1.0
VH12	101	106	1.0	1.0
VH13	956	940	1.0	1.0
VH14	133	134	1.0	1.0
VH15	67	66	1.0	1.0

Continued on next page

Table B.1 – *Continued from previous page*

Tag	Found	Actual	Precision	Recall
VH15[+NEG]	1	1	1.0	1.0
VH16	280	280	1.0	0.98969072164
VH17	56	55	1.0	0.94117647058
VH21	201	199	1.0	1.0
VH22	24	24	1.0	1.0
VI1	28	28	1.0	1.0
VI2	50	50	1.0	1.0
VI3	10	11	1.0	1.0
VJ1	453	452	1.0	0.98930481283
VJ1[+NEG]	1	1	1.0	1.0
VJ2	159	159	1.0	1.0
VJ2[+SPV]	1	1	1.0	1.0
VJ3	552	540	1.0	1.0
VJ3[+NEG]	2	2	1.0	1.0
VK1	775	775	1.0	0.99186991869
VK1[+NEG]	1	1	1.0	1.0
VK2	225	227	1.0	0.98076923076
VK2[+NEG]	1	1	1.0	1.0
VL1	117	117	1.0	1.0
VL2	131	130	1.0	1.0
VL3	1	1	1.0	1.0
VL4	298	297	1.0	1.0
V_11	1116	1035	1.0	0.6
V_12	16	18	1.0	1.0
V_2	707	684	1.0	0.66666666666

Appendix C

Details of PTT Corpus

Table C.1: Details of PTT Corpus

Board	Post	First post	Token	Type
Gossiping (八卦版)	183,838	2005-06-20	27,577,278	453,986
Hate (黑特版)	87,005	2003-04-06	11,262,315	194,603
Food (food 版)	53,629	2003-03-22	6,142,365	132,271
AllTogether (歐兔版)	42,393	2002-07-12	6,807,991	96,465
TaichungBun (台中版)	33,589	2004-07-26	4,076,268	107,317
WomenTalk (女人聊天版)	33,309	2003-04-23	8,460,551	142,644
Kaohsiung (高雄版)	33,124	2008-06-18	3,964,616	118,365
Tainan (台南版)	30,976	2006-02-08	3,217,383	108,677
Hsinchu (新竹版)	28,136	2005-05-02	3,233,960	91,809
StupidClown (笨版)	27,384	2005-04-23	6,320,527	127,035
movie (電影版)	27,229	2004-01-08	9,217,048	198,170
joke (就可版)	27,208	2001-05-14	2,357,648	96,132
FuMouDiscuss (服貿版)	27,041	2014-05-02	4,644,545	113,877
Boy-Girl (男女版)	26,270	2006-12-10	10,559,750	124,232
iPhone (哀鳳版)	25,553	2009-11-15	2,500,198	66,315
NBA (NBA 版)	21,156	2004-05-24	5,417,616	123,320
ChungLi (中壢版)	19,386	2006-09-09	2,089,925	74,907
happy (黑皮版)	19,125	2004-02-29	1,408,294	54,428
PingTung (屏東版)	18,157	2004-11-06	2,051,971	86,131
I-Lan (宜蘭版)	17,621	2004-08-14	2,251,268	86,873
Yunlin (雲林版)	17,605	2005-08-29	1,884,027	78,525
Keelung (基隆版)	17,579	2005-05-25	2,004,838	79,071
Taitung (台東版)	16,681	2005-04-25	2,188,322	82,696
Beauty (表特版)	16,285	2004-07-30	755,986	45,878
Miaoli (苗栗版)	16,062	2005-08-12	1,710,413	71,447
FengYuan (豐原版)	16,048	2007-11-02	1,685,239	71,079
Nantou (南投版)	15,996	2005-12-03	2,032,941	74,431

Continued on next page

Table C.1 – *Continued from previous page*

Board	Post	First post	Token	Type
Taoyuan (桃園版)	15,798	2003-05-08	1,605,410	68,383
ChangHua (彰化版)	15,712	2004-11-21	1,794,450	77,664
Hualien (花蓮版)	15,611	2005-02-03	2,107,496	79,100
TaichungCont (台中市版)	14,592	2005-06-13	1,728,980	72,615
FongShan (鳳山版)	13,987	2005-08-12	1,452,811	64,896
PH-sea (澎湖版)	13,649	2004-06-05	2,773,935	100,583
Jinmen (金門版)	8,255	2005-10-27	1,104,985	60,727
Chiayi (嘉義版)	7,429	2004-06-10	741,446	45,069
Daliao (大寮版)	2,268	2007-01-21	275,278	24,102
Linyuan (林園版)	2,051	2009-12-13	234,834	23,826
Matsu (馬祖版)	1,838	2007-04-07	273,643	21,705

Appendix D

Procedure of `Jseg` segmentation

1. Example

```
sentence = u'波羅的海三小國'
dictionary = u'''
波 772
波羅 2
波羅的海 20
羅 70
的 581859
三 17204
小 8830
國 4406'''

print dictionary
```

```
波 772
波羅 2
波羅的海 20
羅 70
的 581859
三 17204
小 8830
國 4406
```

2. Generate Trie

```
FREQ = {}
TRIE = {}
TOTAL = 0.0
```

```
import json

for line in dictionary.split('\n'):
    word, freq = line.split(' ')
    freq = float(freq)
    FREQ[word] = freq
    TOTAL += freq
    p = TRIE
    for char in word:
        if char not in p:
            p[char] ={}
        p = p[char]
    p['']='' #ending flag


print 'TRIE: ', json.dumps(TRIE, ensure_ascii=False)
print 'FREQ: ', json.dumps(FREQ, ensure_ascii=False)
print 'TOTAL: ', TOTAL
```

```
TRIE:  {'波': {'': '', '羅': {'': '', '的': {'海': {'':
   ''}}}}}, '羅': {'': ''}, '的': {'': ''}, '三': {'':
   ''}, '國': {'': ''}, '小': {'': ''}}
FREQ:  {'波羅': 2.0, '波': 772.0, '羅': 70.0, '的':
   581859.0, '三': 17204.0, '國': 4406.0, '小': 8830.0, '
   波羅的海': 20.0}
TOTAL:  613163.0
```

3. Normalize frequency

```
from math import log

FREQ = [(word,log(float(count)/TOTAL)) for word,count in
   FREQ.iteritems()]
FREQ = dict(FREQ)
min_freq = min(FREQ.itervalues())

print 'Normalied frequency: ', json.dumps(FREQ, ensure_
   ascii=False)
print 'Minimal frequency: ', min_freq
```

```
Normalied frequency:  {'波羅': -12.633238904395194, '波':
    -6.677401534930362, '羅': -9.077890842905779, '的':
   -0.05240265563444316, '三': -3.5734888910515115, '國':
    -4.93566355759285, '小': -4.240475791357133, '
   波羅的海': -10.330653811401147}
Minimal frequency:  -12.6332389044
```

4. Get Directed acyclic graph (DAG)

```
senlen = len(sentence)
i,j=0,0
p = TRIE
DAG = {}
while i<senlen:
    c = sentence[j]
    if c in p:
        p = p[c]
        if '' in p:
            if i not in DAG:
                DAG[i]=[]
            DAG[i].append(j)
        j+=1
        if j>=senlen:
            i+=1
            j=i
            p=TRIE
    else:
        p = TRIE
        i+=1
        j=i
for i in xrange(len(sentence)):
    if i not in DAG:
        DAG[i] =[i]
print 'DAG: ', DAG
```

```
DAG:  {0: [0, 1, 3], 1: [1], 2: [2], 3: [3], 4: [4], 5:
   [5], 6: [6]}
```

5. Calculate DAG

```
route = {}

route[senlen] = (0.0,'')
for idx in xrange(senlen-1,-1,-1):
    candidates = [ ( FREQ.get(sentence[idx:x+1],min_freq)
        + route[x+1][0],x ) for x in DAG[idx] ]
    route[idx] = max(candidates)
import pprint
print 'route: '
pprint.pprint(route)
```

```
route:
```

```
{0: (-23.08028205140264, 3),
 1: (-34.51316064293691, 1),
 2: (-25.43526980003113, 2),
 3: (-25.382867144396688, 3),
 4: (-12.749628240001494, 4),
 5: (-9.176139348949983, 5),
 6: (-4.93566355759285, 6),
 7: (0.0, '')}
```

6. HMM

```
MIN_FLOAT=-3.14e100

PrevStatus = {
              'B':('E','S'),
              'M':('M','B'),
              'S':('S','E'),
              'E':('B','M')
              }
```

7. Probability

```
initP = {
     'B': -0.26268660809250016,
     'E': -3.14e+100,
     'M': -3.14e+100,
     'S': -1.4652633398537678
     }

# math.exp('B') = 0.7689828525554734
# math.exp('E') = 0
# math.exp('M') = 0


# transition probability
tranP={
   'B': {'E': -0.51082562376599000, 'M':
      -0.9162907318741550},
   'E': {'B': -0.58971497368545130, 'S':
      -0.8085250474669937},
   'M': {'E': -0.33344856811948514, 'M':
      -1.2603623820268226},
   'S': {'B': -0.72119656546698410, 'S':
      -0.6658631448798212}
```

```
   }

# emission probability
emisP = {
     'B': {u'小': -5.795450196520336},
     'M': {u'小': -7.367972609622759},
     'E': {u'小': -5.095185283088565},
     'S': {u'小': -6.247496378368061}
     }
```

8. Viterbi

```
def viterbi(obs, states, start_p, trans_p, emit_p):
    V = [{}] #tabular
    path = {}
    for y in states: #init
        V[0][y] = start_p[y] + emit_p[y].get(obs[0],MIN_
           FLOAT)
        path[y] = [y]
    for t in range(1,len(obs)):
        V.append({})
        newpath = {}
        for y in states:
            em_p = emit_p[y].get(obs[t],MIN_FLOAT)
            (prob,state ) = max([(V[t-1][y0] + trans_p[y
               0].get(y,MIN_FLOAT) + em_p ,y0) for y0 in
               PrevStatus[y] ])
            V[t][y] =prob
            newpath[y] = path[state] + [y]
        path = newpath

    (prob, state) = max([(V[len(obs) - 1][y], y) for y in
        ('E','S')])

    return (prob, path[state])

res = viterbi(u'波羅的海三小國', ('B', 'M', 'E', 'S'),
   initP, tranP, emisP)
print res
```

```
(-1.884e+101, ['S', 'S', 'S', 'S', 'S', 'S', 'S'])
```

9. HMM Segmentation

```
def hmm_seg(sentence):
    con = []
    global emit_P
    prob, pos_list =  viterbi(sentence,('B','M','E','S'),
        initP, tranP, emisP)
    begin, next = 0,0
    for i,char in enumerate(sentence):
        pos = pos_list[i]
        if pos=='B':
            begin = i
        elif pos=='E':
            con.append(sentence[begin:i+1])
            next = i+1
        elif pos=='S':
            con.append(char)
            next = i+1
    if next<len(sentence):
        con.append(sentence[next:])

    return con

res = hmm_seg(sentence)
print json.dumps(res, ensure_ascii=False)
```

['波', '羅', '的', '海', '三', '小', '國']

10.Cut DAG

```
def cut_DAG(sentence):
    dag_con = []
    x = 0
    buf =u''
    N = len(sentence)
    while x<N:
        y = route[x][1]+1
        l_word = sentence[x:y]
        if y-x==1:
            buf+= l_word
        else:
            if len(buf)>0:
                if len(buf)==1:
                    dag_con.append(buf)
                    buf=u''
                else:
                    if (buf not in FREQ):
                        regognized = hmm_seg(buf)
                        for t in regognized:
```

```
                            dag_con.append(t)
                    else:
                        for elem in buf:
                            dag_con.append(elem)
                    buf=u''
            dag_con.append(l_word)
        x =y

    if len(buf)>0:
        if len(buf)==1:
            dag_con.append(buf)
        else:
            if (buf not in FREQ):
                regognized = hmm_seg(buf)
                for t in regognized:
                    dag_con.append(t)
            else:
                for elem in buf:
                    dag_con.append(elem)

    return dag_con

dag_con = cut_DAG(sentence)
print json.dumps(dag_con, ensure_ascii=False)
```

```
['波羅的海', '三', '小', '國']
```

11.Regular expression

```
import re
re_han = re.compile(ur'([\u4E00-\u9FA5a-zA-Z0-9+#&\._]+)
   ', re.U)
re_skip = re.compile(ur'(\r\n|\s)', re.U)

blocks = re_han.split(sentence)
print json.dumps(blocks, ensure_ascii=False)
```

```
['', '波羅的海三小國', '']
```

12.Finalize

```
con = []
for blk in blocks:
    if len(blk)==0:
        continue
```

```
    if re_han.match(blk):
        for word in cut_DAG(blk):
            con.append(word)
    else:
        tmp = re_skip.split(blk)
        for x in tmp:
            if re_skip.match(x):
                con.append(x)
            elif not cut_DAG:
                for xx in x:
                    con.append(xx)
            else:
                con.append(x)

print json.dumps(con, ensure_ascii=False)
```

['波羅的海', '三', '小', '國']

Appendix E

CKIP POS

Table E.1: CKIP POS

POS	Description
A	非謂形容詞
Caa	對等連接詞，如：和、跟
Cab	連接詞，如：等等
Cbab	連接詞，如：的話
Cbaa, Cbba, Cbbb, Cbca, Cbcb	關聯連接詞
Daa	數量副詞
Dfa	動詞前程度副詞
Dfb	動詞後程度副詞
Di	時態標記
Dk	句副詞
Dab, Dbaa, Dbab, Dbb, Dbc, Dc, Dd, Dg, Dh, Dj	副詞
Naa, Nab, Nac, Nad, Naea, Naeb	普通名詞
Nba, Nbc	專有名稱
Nca, Ncb, Ncc, Nce	地方詞
Ncda, Ncdb	位置詞
Ndaa, Ndab, Ndc, Ndd	時間詞
Neu	數詞定詞
Nes	特指定詞
Nep	指代定詞
Neqa	數量定詞
Neqb	後置數量定詞
Nfa, Nfb, Nfc, Nfd, Nfe, Nfg, Nfh, Nfi	量詞
Ng	後置詞
Nhaa, Nhab, Nhac, Nhb, Nhc	代名詞
Nv1,Nv2,Nv3,Nv4	名物化動詞
I	感嘆詞
P*	介詞
Ta, Tb, Tc, Td	語助詞

Continued on next page

Table E.1 – *Continued from previous page*

POS	Description
VA11,12,13,VA3,VA4	動作不及物動詞
VA2	動作使動動詞
VB11,12,VB2	動作類及物動詞
VC2, VC31,32,33	動作及物動詞
VC1	動作接地方賓語動詞
VD1, VD2	雙賓動詞
VE11, VE12, VE2	動作句賓動詞
VF1, VF2	動作謂賓動詞
VG1, VG2	分類動詞
VH11,12,13,14,15,17,VH21	狀態不及物動詞
VH16, VH22	狀態使動動詞
VI1,2,3	狀態類及物動詞
VJ1,2,3	狀態及物動詞
VK1,2	狀態句賓動詞
VL1,2,3,4	狀態謂賓動詞
V_2	有

References

Bird, Steven, Ewan Klein & Edward Loper. (2009). *Natural language processing with python*. ” O’Reilly Media, Inc.”.

Brill, Eric. (1992). A simple rule-based part of speech tagger. In *Proceedings of the workshop on speech and natural language*, 112–116. Association for Computational Linguistics.

Chen, Feng-Yi, Pi-Fang Tsai, Keh-Jiann Chen & Chu-Ren Huang. (1999). The construction of sinica treebank. *Computational Linguistics and Chinese Language Processing* 4. 87–104.

Chen, Keh-Jiann, Chu-Ren Huang, Li-Ping Chang & Hui-Li Hsu. (1996). Sinica corpus: Design methodology for balanced corpora. *Language* 167. 176.

Chen, Mei-Yu, Hsin-Ni Lin, Chang-An Shih, Yen-Ching Hsu, Pei-Yu Hsu & Shu-Kai Hsieh. (2010). Classifying mood in plurks. In *Rocling*, .

Christ, Oliver. (1994). The ims corpus workbench technical manual. *Institut für maschinelle Sprachverarbeitung, Universit at Stuttgart* .

Christ, Oliver, Bruno M Schulze, Anja Hofmann & Esther Koenig. (1999). The ims corpus workbench: Corpus query processor (cqp): User’s manual. *University of Stuttgart* 8.

Crystal, David. (2011). *Dictionary of linguistics and phonetics*, vol. 30. John Wiley & Sons.

De Schryver, Gilles-Maurice. (2002). Web for/as corpus: A perspective for the african languages. *Nordic Journal of African Studies* 11(2). 266–282.

Evert, Stefan. (2005). The cqp query language tutorial .

Evert, Stefan & Andrew Hardie. (2011). Twenty-first century corpus workbench: Updating a query architecture for the new millennium .

Fairon, Cédrick, Kévin Macé & Hubert Naets. (2008). Glossanet 2: a linguistic search engine for rss-based corpora. In *Proceedings of the 4th web as corpus workshop (wac-4)*, 34–39. Citeseer.

Fletcher, William H. (2004). Making the web more useful as a source for linguistic corpora. *Language and Computers* 52(1). 191–205.

Fletcher, William H. (2006). Concordancing the web: promise and problems, tools and techniques. *Language and Computers* 59(1). 25–45.

Hayashi, Yoshihiko, Gen' ichiro Kikui & Seiji Susaki. (1997). Titan: A cross-linguistic search engine for the www. In *Working notes of aaai-97 spring symposiums on cross-language text and speech retrieval*, 58–65.

Hsieh, Shu-Kai. (2014). Evaluating chinese web-as-corpus, Linguistic Corpus and Corpus Linguistics in the Chinese Context.

Huang, Chu-Ren, Adam Kilgarriff, Yiching Wu, Chih-Ming Chiu, Simon Smith, Pavel Rychly, Ming-Hong Bai & Keh-Jiann Chen. (2005). Chinese sketch engine and the extraction of grammatical collocations. In *Proceedings of the fourth sighan workshop on chinese language processing*, 48–55.

Jalilvand, Abbas & Naomie Salim. (2012). Sentiment classification using graph based word sense disambigution. In *Advanced machine learning technologies and applications*, 351–358. Springer.

Kaplan, Andreas M. & Michael Haenlein. (2010). Users of the world, unite! the challenges and opportunities of social media. *Business Horizons* 53(1). 59 – 68. doi: http://dx.doi.org/10.1016/j.bushor.2009.09.003. http://www.sciencedirect.com/science/article/pii/S0007681309001232.

Kilgarriff, Adam. (2001). Web as corpus. In *Proceedings of corpus linguistics 2001*, 342–344. Corpus Linguistics. Readings in a Widening Discipline.

Kilgarriff, Adam & Gregory Grefenstette. (2003). Introduction to the special issue on the web as corpus. *Computational linguistics* 29(3). 333–347.

Kilgarriff, Adam & Iztok Kosem. (2012). Corpus tools for lexicographers. *Granger, S. and M. Paquot (Eds.)* 2012. 31–55.

Kilgarriff, Adam & David Tugwell. (2002). Sketching words. *Lexicography and Natural Language Processing: A Festschrift in Honour of B. TS Atkins* 125–137.

Ku, Lun-Wei & Hsin-Hsi Chen. (2007). Mining opinions from the web: Beyond relevance retrieval. *Journal of the American Society for Information Science and Technology* 58(12). 1838–1850.

Leech, Geoffrey. (2006). New resources, or just better old ones? the holy grail of representativeness. *Language and Computers* 59(1). 133–149.

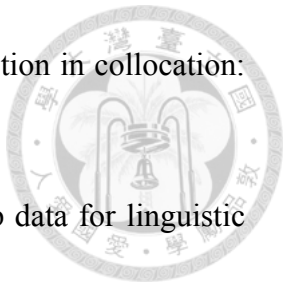
Lin, Qian-Xiang & Chia-Hui Chang. (2006). 基於特製隱藏式馬可夫模型之中文斷詞研究 (chinese word segmentation using specialized hmm)[in chinese]. 中央大學資訊工程學系學位論文 1–41.

Liu, Tsun-Jui. (May 2014). Gei ta in taiwanese mandarin: A corpus-based study. Paper presented at the 2014 National Conference on Linguistics, 23-24th May, Tunghai University, Taiwan.

Liu, Tsun-Jui, Shu-Kai Hsieh & Laurent Prevot. (2013). Observing features of ptt neologisms: A corpus-driven study with n-gram model. In *Rocling*, Association for Computational Linguistics and Chinese Language Processing (ACLCLP), Taiwan.

Liu, Vinci & James R Curran. (2006). Web text corpus for natural language processing. In *Eacl*, .

Liu, Yu-Ming. (2012). *A study for the culture of university student's buzzwords phenomenon on bbs in taiwan*. National Chi Nan University MA thesis.



Lu, Pei-Yu, Yu-Yun Chang & Shu-Kai Hsieh. (2013). Causing emotion in collocation: An exploratory data analysis. In *Rocling*, .

Lüdeling, Anke, Stefan Evert & Marco Baroni. (2006). Using web data for linguistic purposes. *Language and Computers* 59(1). 7–24.

Manning, Christopher D & Hinrich Schütze. (1999). *Foundations of statistical natural language processing*. MIT press.

McCreadie, Richard, Ian Soboroff, Jimmy Lin, Craig Macdonald, Iadh Ounis & Dean McCullough. (2012). On building a reusable twitter corpus. In *Proceedings of the 35th international acm sigir conference on research and development in information retrieval*, 1113–1114. ACM.

McEnery, Tony & Andrew Wilson. (2001). *Corpus linguistics: An introduction*. Edinburgh University Press 2nd edn.

Perkins, Jacob. (2011). Nltk-trainer [software]. Available from: http:// nltk-trainer.readthedocs.org/.

Petrovic, Sasa, Miles Osborne & Victor Lavrenko. (2010). The edinburgh twitter corpus. In *Proceedings of the naacl hlt 2010 workshop on computational linguistics in a world of social media*, 25–26.

Ptaszynski, Michal, Jacek Maciejewski, Pawel Dybala, Rafal Rzepka & Kenji Araki. (2010). Cao: A fully automatic emoticon analysis system based on theory of kinesics. *Affective Computing, IEEE Transactions on* 1(1). 46–59.

Renouf, Antoinette. (2003). Webcorp: providing a renewable data source for corpus linguists. *Language and Computers* 48(1). 39–58.

Rychlỳ, Pavel. (2008). A lexicographer-friendly association score. *Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN* 6–9.

Shivhare, Shiv Naresh & Saritha Khethawat. (2012). Emotion detection from text. *arXiv preprint arXiv:1205.4944* .

Su, Hsi-Yao. (2003). The multilingual and multi-orthographic taiwan-based internet: Creative uses of writing systems on college-affiliated bbss. *Journal of Computer-Mediated Communication* 9(1). 0–0. doi:10.1111/j.1083-6101.2003.tb00357.x. http://dx.doi.org/10.1111/j.1083-6101.2003.tb00357.x.

Sun, Junyi. (2012). Jieba [software]. Available from: https://github.com/fxsjy/jieba.

Volk, Martin. (2002). Using the web as corpus for linguistic research. *Catcher of the Meaning. Pajusalu, R., Hennoste, T.(Eds.). Dept. of General Linguistics* 3.