行政院國家科學委員會補助專題研究計畫成果報告

機率式建模技術與自然語言的標記、認知和教學

計畫類別：■個別型計畫　□整合型計畫
計畫編號：NSC 97－2221－E－004－007－MY2
執行期間： 97 年 8 月　1 日至　100 年 1 月　31 日

執行機構及系所：國立政治大學資訊科學系

計畫主持人：劉昭麟
共同主持人：蔡介立(國立政治大學心理系)、高照明(國立台灣大學外語系)
計畫參與人員：賴敏華、田侃文、黃志斌、黃昭憲、莊怡軒、陳建良、
張裕淇、蔡家琦、王瑞平

成果報告類型(依經費核定清單規定繳交)：□精簡報告　■完整報告
(為避免重複，期中報告已經呈報之成果將不再贅述)

本計畫除繳交成果報告外，另須繳交以下出國心得報告：
□赴國外出差或研習心得報告
■赴大陸地區出差或研習心得報告
■出席國際學術會議心得報告
□國際合作研究計畫國外研究報告

處理方式：除列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

中　華　民　國　100 年　4 月　1 日

國科會補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估

■ 達成目標

說明：

延續之前的相關研究，本研究案完成了一項以機率模型建構學生學習模型的研究，並且發表一系列之相關學術會議論文和包含一篇 IJAIED 之兩篇長篇期刊論文。在認知與教學方面，藉由此研究案的資助，我們陸續發表於國際間卓俱聲望的學術研討會，例如 ACL 和 COLING。

2. 研究成果在學術期刊發表或申請專利等情形：

論文：■已發表 □未發表之文稿 □撰寫中 □無

專利：□已獲得 □申請中 ■無

技轉：□已技轉 □洽談中 ■無

其他：（以 100 字為限）

主要論文發表如下。

1. C.-L. Liu. Selecting Bayesian-network models based on simulated expectation, *Behaviormetrika*, **36**(1), 1–25. 2009.
2. C.-L. Liu. A simulation-based experience in learning structures of Bayesian networks to represent how students learn composite concepts, *International Journal of Artificial Intelligence in Education*, **18**(3), 237–285. 2008.
3. C.-L. Liu, M.-H. Lai, Y.-H. Chuang, and C.-Y. Lee. Visually and phonologically similar characters in incorrect simplified Chinese words, *Proceedings of the Twenty Third International Conference on Computational Linguistics* (COLING'10), posters, 739–747. 2010.
4. C.-L. Liu, K.-W. Tien, M.-H. Lai, Y.-H. Chuang, and S.-H. Wu. Capturing errors in written Chinese words, *Proceedings of the Forty Seventh Annual Meeting of the Association for Computational Linguistics* (ACL'09), short papers, 25–28. 2009.

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）（以 500 字為限）

本研究於 2007 年底提案之時，即已著手進行以機率式模型建構學生學習模型之相關研究，而且也正著手進行關於漢字錯字分析的基礎工作。經過兩年的努力之後，我們完成了以機率式模型建構學生學習模型的探索，瞭解了其中的困難之處，並且將所得的經驗發表於一系列之國際學術會議以及兩篇國際學術期刊論文之中。

語言教學方面，我們從 2007 年的初步探索，歷經 2008 年和 2009 年多位同學的持續努力，我們完成了以倉頡詳馬來捕捉形體相近漢字的相關研究，並且完成了數個實際的應用，過程中的相關成果均發表於 ACL 年會之中。至 2010 年，我們將相關研究經驗擴大到簡體漢字，相關結果也發表於 COLING 年會。

在認知科技方面，我們的成果仍屬初階，目前只獲得少量的眼動資料，雖然已經啟動相關研究，但是仍然沒有建立精確的眼動軌跡與漢字閱讀歷程的關係。這一部份的成果現在只發表在 2010 年 TAAI 年會的國際議程中。我們希望能夠在這一方面繼續努力。

除了發表論文之外，這一研究計畫對於社會最實際的貢獻在於搭配 2010 到 2011 年的後續計畫中，我們實際地建立了一個可以輔助孩童學習漢字的軟體，並且實際在台北市某國小實地測試，而且證明該軟體對於學習成效的明顯助益。

由於計畫主持人於 2010 年中出國短期訪問直到 2011 年一月初才返國，基於須要準備不少工作，並且有一些計畫工作的延遲，所以計畫工作遲到 2010 年下半年方才結束，導致報帳與相關結案作業均相當延遲。

中文摘要

本研究案於撰寫計畫申請書時，預計進行兩大類工作。第一類是利用貝氏網路來建構學生學習歷程的模型，第二類工作則是利用計算語言學處理自然語言的技術，來開發可以協助電腦輔助語文教學的軟體。以下我們分類說明這兩年期的計畫中，我們所發表的 19 篇論文。雖然我們著墨的應用領域不少，但是都是基於應用人工智慧與計算語言學的相關技術來作有用的應用。

經過兩年的努力，雖然我們又發掘了更多先前沒有想到且值得深入的議題，但是我們相信我們的工作目標已經完成相當高的比例。在過去兩年之間，我們共發表三篇國際學術期刊論文，六篇國際學術會議論文與十篇國內學術會議論文。比較具有代表性的長篇期刊論文發表於 *International Journal of Artificial Intelligence in Education* (*IJAIED*)，此外我們也在 2009 發表了一篇 ACL 短文論文、在 2010 年發表的一篇 COLING 的海報論文。雖然這樣的結果顯然不是最佳的可能成果，不過希望可以算是不錯的成果。

除了 *IJAIED* 論文與另一篇發表於 *Behaviormetika* 的論文之外，這兩年論文主要的議題都是關於應用計算語言學來協助建構電腦輔助教學的軟體的議題。我們主要集中於研究包含簡體字和繁體字的中文錯字。錯字的來源可能源於發音相近的字，也可能源於字形相近的字。我們延伸的倉頡碼的原始概念，建立了倉頡詳碼，然後應用倉頡詳碼的概念來計算中文近形字。再輔以網路搜尋技術的協助，我們發現一個很好的方法可以複製人們常寫錯的漢字。在這樣的基礎之上，我們開發了兩個應用軟體，一個是協助老師出中文改錯字試題的軟體，另一個則是輔助小學生學習漢字的遊戲軟體。這一方面的論文發表於 IC1、IC2、IC4、IC5、IC6 和 J1(請參閱本報告中之論文列表)。基於這一些國際會議的論文的整合結果，在進一步利用 2010 年七月之後的資助之下，我們已經整理成期刊論文，並且被 ACM 的 *Transaction on Asian Language Information Processing* 接受發表。DC1 的內容，則是利用過去計算相近漢字的技術，用來輔助心理語言學專家建構一個有助於學生學習漢字的遊戲軟體。

在學生學習歷程的機率式建模和中文錯字之外，我們還有餘力作更多的相關研究。IC3 和 DC7 是關於如何運用自然語言處理技術，來協助教師編輯句子重組試題的研究成果。DC2 和 DC5 是利用語句分析技術來探討閱讀測驗難度的問題。

DC4 是關於如何利用文字處理技術來猜測中文短句中的情緒。

DC3、DC6 和 DC8 都是關於機器翻譯的相關技術。DC3 則是偏向於如何利用機器翻譯技術來輔助英文試題的中譯工作。

DC9 是延續多年以前關於中文訴訟文書的國科會研究計畫的末尾結束工作。

DC10 則是一個電腦遊戲的設計。這樣的發展有助於我們累積發展輔助語文學習的遊戲設計。

在人員訓練方面，本計畫執行期間合計畢業七位碩士班研究生，目前均順利就業或者服役中。

英文摘要

At the time the proposal for the research project was drafted, we planned two types of research work. The first was to apply Bayesian networks as the language for modeling students' learning procedures. The second one was to employ techniques of natural language processing for assisting the construction of software for computer-assisted language learning. After two years of work, we have published 19 papers based on our findings. Although these papers cover a wide range of applications, they share the same technical foundations: artificial intelligence and computational linguistics.

After two years of work, we are afraid that we have encountered more challenges than we thought, and we have learned a lot from this experience. Despite the challenges, we have worked hard enough to find some useful results. The results were published in three international journal articles, six international academic conferences papers, and 10 domestic academic conferences papers. Among these publications, we thought the long article published in the *International Journal of Artificial Intelligence in Education* (*IJAIED*) is quite representative. In addition, we published our results, as short papers, in the most prestigious conferences in the field of Computational Linguistics, i.e., ACL 2009 and COLING 2010. Although we cannot say that such results are perfect, we hope that these accomplishments are reasonable for the resources that we were granted.

The papers that we published in *IJAIED* and *Behaviormetrika* were about student modeling with Bayesian networks. The target problem was about modeling the learning procedure with which students learn composite concepts.

At almost the same time that we finished the work on student modeling, we began to work on the issues related to incorrect Chinese characters in text. The majority of our recent publications surround this research topic, and we tried to work on both traditional and simplified Chinese characters. Both intuition and our experience show that the sources of incorrect characters are related to phonological and visual similarity between Chinese characters. In order to enable computers to find visually similar characters, we extend the original Cangjie codes to capture the detailed structures of Chinese characters. Some similarity measures were proposed for the task, and experimental results have proven their effectiveness. With the help of Web-based statistics that we collected from portals like Google, we were able to reproduce incorrect words at very high rate.

Based on such a computing tool, we have built a system that can help teachers to compile test items for word corrections of Chinese characters. We have also employed the tool to build a game-based learning environment to help students learn to read Chinese characters.

For the research direction about incorrect Chinese characters, we have published papers IC1, IC2, IC4, IC5, IC6, and J1 (please refer to the papers listed below). In addition, with the support of an NSC project that began from August 2010, we have actually extended the work reported in the conference papers, and have a journal paper accepted by the ACM *Transaction on Asian Language Information Processing*. A portion of our experience in building a game-based environment for learning Chinese characters was reported in DC1. This game was based on research results of psycholinguistics, and the latest experiments, those we conducted in 2011, proved the effectiveness of the games.

We have conducted some more exploration that we did not mention in the original proposal. We applied techniques for sentence manipulation to help teachers build test items for scrambled sentences, and the results were reported in IC3 and DC7. We tried to learn

about how the texts for reading comprehensions were chosen by analyzing the linguistic structures of the texts, and the results were reported in DC2 and DC5.

DC4 reported how we may infer the emotion carried by short Chinese sentences.

DC3, DC6, and DC8 were related to techniques of machine translation (MT). DC3, in particular, was related how we may apply MT techniques for translating English test items into their Chinese counterparts.

DC9 was a piece of work that related to a previous NSC research project of ours. It reported an information retrieval system for Chinese indictment documents.

DC10 provided our experience in using machine learning techniques in game design. It was one of our first attempt to build a game, and was educational for us to build the game for learning Chinese characters.

中文關鍵詞

貝氏網路、機率式建模、學生學習歷程、中文錯字、改錯字試題、句子重組、心理語言學、中文字學習遊戲

英文關鍵詞

Bayesian Networks, Probabilistic Modeling, Learning Style Modeling, Incorrect Chinese Characters, Word Correction Tests, Scrambled Sentence Tests, Psycholinguistics, Games for Learning Chinese Characters

報告內容：包括前言、研究目的、文獻探討、研究方法、結果與討論（含結論與建議）…等。

由於本研究案發表許多論文，如果要瞭解個別研究工作的參考文獻、研究方法與結果等細節，煩請參考下面表列之各論文。個別計畫的工作內容與關連性已經在前面的摘要中概述。

計畫執行期間之內所發表之論文

期刊論文

J1. Chao-Lin Liu, Jen-Hsiang Lin, and Yu-Chun Wang. Applications of NLP techniques to computer-assisted authoring of test items for elementary Chinese, *US-China Education Review*, **7**(3), 42−52. David Publishing Company, USA, March 2010.

J2. Chao-Lin Liu. Selecting Bayesian-network models based on simulated expectation, *Behaviormetrika*, **36**(1), 1−25. The Behaviormetric Society of Japan, Japan, April 2009.

J3. Chao-Lin Liu. A simulation-based experience in learning structures of Bayesian networks to represent how students learn composite concepts, *International Journal of Artificial Intelligence in Education*, **18**(3), 237−285. IOS Press, The Netherlands, September 2008.

國際學術會議論文

IC1. Shih-Hung Wu, Yong-Zhi Chen, Ping-Che Yang, Tsun Ku, and Chao-Lin Liu. Reducing the false alarm rate of Chinese character error detection and correction, *Proceedings of the First CIPS-SIGHAN Joint Conference on Chinese Language Processing* (CLP'10), 54−61. Beijing, China, 28-29 August 2010.

IC2. Chao-Lin Liu, Min-Hua Lai, Yi-Hsuan Chuang, and Chia-Ying Lee. Visually and phonologically similar characters in incorrect simplified Chinese words, *Proceedings of the Twenty Third International Conference on Computational Linguistics* (COLING'10), posters, 739−747. Beijing, China, 23-27 August 2010.

IC3. Chao-Lin Liu, Chih-Bin Huang, Ying-Tse Sun, and Wei-Ti Kuo. Computer assisted creation of items for scrambled sentence tests, *Proceedings of the Seventeenth International Conference on Computers in Education* (ICCE'09), 117−121. Hong Kong, China, 30 November-4 December 2009.

IC4. Chao-Lin Liu, Kan-Wen Tien, Min-Hua Lai, Yi-Hsuan Chuang, and Shih-Hung Wu. Phonological and logographic influences on errors in written Chinese words, *Proceedings of the Seventh Workshop on Asian Language Resources* (ALR7), the Forty Seventh Annual Meeting of the Association for Computational Linguistics (ACL'09), 84−91. Singapore, 2-7 August 2009.

IC5. Chao-Lin Liu, Kan-Wen Tien, Min-Hua Lai, Yi-Hsuan Chuang, and Shih-Hung Wu. Capturing errors in written Chinese words, *Proceedings of the Forty Seventh Annual Meeting of the Association for Computational Linguistics* (ACL'09), short papers, 25−28. Singapore, 2-7 August 2009.

IC6. Chao-Lin Liu, Kan-Wen Tien, Yi-Hsuan Chuang, Chih-Bin Huang, and Juei-Yu Weng. Two applications of lexical information to computer-assisted item authoring for elementary Chinese, Lecture Notes in Computer Science 5579: *Proceedings of the Twenty Second International Conference on Industrial Engineering & Other Applications of Applied Intelligent Systems* (IEA/AIE '09), 470−480. Tainan, Taiwan, 24-27 June 2009.

國內學術會議論文

DC1. Chia-Ling Lee, Yu-Chi Chang, Chia-Ying Lee, and Chao-Lin Liu. 結合認知理論之電腦輔助漢字教學遊戲, *Proceedings of the 2010 Taiwan Academic Network Conference* (TANET'10), CD-ROM. Tainan, Taiwan, 27-29 October, 2010. (in Chinese)

DC2. Chao-Shainn Huang, Wei-Ti Kuo, Chia-Ling Lee, Chia-Chi Tsai, and Chao-Lin Liu. 以語文特徵為基之中學閱讀測驗短文分級 (Using linguistic features to classify texts for reading comprehension tests at the high school levels), *Proceedings of the Twenty Second Conference on Computational Linguistics and Speech Processing* (ROCLING XXII), 98−112. Nantou, Taiwan, 1-2 September 2010. (in Chinese)

DC3. Chao-Shainn Huang, Yu-Chi Chang, Chao-Lin Liu, and Yuen-Hsien Tseng. 以共現資訊為基礎增進中學英漢翻譯試題與解答之詞彙對列 (Using co-occurrence information to improve Chinese-English word alignment in translation test items for high school students), *Proceedings of the Twenty Second Conference on Computational Linguistics and Speech Processing* (ROCLING XXII), 128−142. Nantou, Taiwan, 1-2 September 2010. (in Chinese)

DC4. Ying-Tse Sun, Chien-Liang Chen, Chun-Chieh Liu, Chao-Lin Liu, and Von-Wun Soo. 中文短句之情緒分類 (Sentiment classification of short Chinese sentences), *Proceedings of the Twenty Second Conference on Computational Linguistics and Speech Processing* (ROCLING XXII), 184−198. Nantou, Taiwan, 1-2 September 2010. (in Chinese)

DC5. Wei-Ti Kuo, Chao-Shainn Huang, Min-Hua Lai, Chao-Lin Liu, and Zhao-Ming Gao. 適用於中學英文閱讀測驗短文分類的特徵比較, *Proceedings of the Fourteenth Conference on Artificial Intelligence and Applications* (TAAI'09), CD-ROM. Taichung, Taiwan, 30-31 October 2009. (in Chinese)

DC6. Kan-Wen Tien, Yuen-Hsien Tseng, and Chao-Lin Liu. 中英文專利文書之文句對列(Sentence alignment of English and Chinese patent documents), *Proceedings of the Twenty First Conference on Computational Linguistics and Speech Processing* (ROCLING XXI), 85−99. Taichung, Taiwan, 1-2 September 2009. (in Chinese)

DC7. Chih-Bin Huang, Chao-Lin Liu, Wei-Ti Kuo, Ying-Tse Sun, and Min-Hua Lai. 電腦輔助句子重組試題編製(Computer assisted test-item generation for sentence reconstruction), *Proceedings of the Twenty First Conference on Computational Linguistics and Speech Processing* (ROCLING XXI), 165−178. Taichung, Taiwan, 1-2 September 2009. (in Chinese)

DC8. Yuen-Hsien Tseng, Chao-Lin Liu, and Ze-Jing Chuang. 專利雙語語料之中、英對照詞自動擷取 (Automatic term pair extraction from bilingual patent corpus), *Proceedings of the Twenty First Conference on Computational Linguistics and Speech Processing* (ROCLING XXI), 279−292. Taichung, Taiwan, 1-2 September 2009. (in Chinese)

DC9. Chia-Liang Lan, Min-Hua Lai, Kan-Wen Tien and Chao-Lin Liu. 訴訟文書檢索系統, *Proceedings of the Thirteenth Conference on Artificial Intelligence and Applications* (TAAI'08), 305−312. Yilan, Taiwan, 21-22 November 2008. (in Chinese)

DC10. Zeng-Hong Lin and Chao-Lin Liu. 任意棋盤的 Othello 遊戲, *Proceedings of the Thirteenth Conference on Artificial Intelligence and Applications* (TAAI'08), 443−449. Yilan, Taiwan, 21-22 November 2008. (in Chinese)

在期中報告中我們已經附上當時已經發表的論文，以下我們再附上兩篇更新的論文。分別是前面論文表列中的 IC2 和 IC3。

Visually and Phonologically Similar Characters in Incorrect Simplified Chinese Words

Chao-Lin Liu[†] Min-Hua Lai[‡] Yi-Hsuan Chuang[↑] Chia-Ying Lee[↕]
[†‡↑]Department of Computer Science; [†↕]Center for Mind, Brain, and Learning
National Chengchi University
[↕]Institute of Linguistics, Academia Sinica
{[†]chaolin, [‡]g9523, [↑]g9804}@cs.nccu.edu.tw, [↕]chiaying@gate.sinica.edu.tw

Abstract

Visually and phonologically similar characters are major contributing factors for errors in Chinese text. By defining appropriate similarity measures that consider extended Cangjie codes, we can identify visually similar characters within a fraction of a second. Relying on the pronunciation information noted for individual characters in Chinese lexicons, we can compute a list of characters that are phonologically similar to a given character. We collected 621 incorrect Chinese words reported on the Internet, and analyzed the causes of these errors. 83% of these errors were related to phonological similarity, and 48% of them were related to visual similarity between the involved characters. Generating the lists of phonologically and visually similar characters, our programs were able to contain more than 90% of the incorrect characters in the reported errors.

1 Introduction

In this paper, we report the experience of our studying the errors in simplified Chinese words. Chinese words consist of individual characters. Some words contain just one character, but most words comprise two or more characters. For instance, "卖" (mai4)[1] has just one character, and "语言" (yu3 yan2) is formed by two characters. Two most common causes for writing or typing incorrect Chinese words are due to visual and phonological similarity between the correct and the incorrect characters. For instance, one might use "划" (hwa2) in the place of "画"(hwa4) in "刻画形象" (ke1 hwa4 xing2 xiang4) partially because of phonological similarity; one might replace "拙" (zhuo2) in "心劳力拙" (xin1 lao2 li4 zhuo2) with "绌" (chu4) partially due to visual similarity. (We do not claim that the visual or phonological similarity alone can explain the observed errors.)

Similar characters are important for understanding the errors in both traditional and simplified Chinese. Liu et al. (2009a-c) applied techniques for manipulating correctness of Chinese words to computer assisted test-item generation. Research in psycholinguistics has shown that the number of neighbor characters influences the timing of activating the mental lexicon during the process of understanding Chinese text (Kuo et al. 2004; Lee et al. 2006). Having a way to compute and find similar characters will facilitate the process of finding neighbor words, so can be instrumental for related studies in psycholinguistics. Algorithms for optical character recognition for Chinese and for recognizing written Chinese try to guess the input characters based on sets of confusing sets (Fan et al. 1995; Liu et al., 2004). The confusing sets happen to be hand-crafted clusters of visually similar characters.

It is relatively easy to judge whether two characters have similar pronunciations based on their records in a given Chinese lexicon. We will discuss more related issues shortly.

To determine whether two characters are visually similar is not as easy. Image processing techniques may be useful but is not perfectly feasible, given that there are more than fifty thousand Chinese characters (HanDict, 2010) and that many of them are similar to each other in special ways. Liu et al. (2008) extend the Cangjie codes (Cangjie, 2010; Chu, 2010) to encode the layouts and details about traditional

[1] We show simplified Chinese characters followed by their Hanyu pinyin. The digit that follows the symbols for the sound is the tone for the character.

Chinese characters for computing visually similar characters. Evidence observed in psycholinguistic studies offers a cognition-based support for the design of Liu et al.'s approach (Yeh and Li, 2002). In addition, the proposed method proves to be effective in capturing incorrect traditional Chinese words (Liu et al., 2009a-c).

In this paper, we work on the errors in simplified Chinese words by extending the Cangjie codes for simplified Chinese. We obtain two lists of incorrect words that were reported on the Internet, analyze the major reasons that contribute to the observed errors, and evaluate how the new Cangjie codes help us spot the incorrect characters. Results of our analysis show that phonological and visual similarities contribute similar portions of errors in simplified and traditional Chinese. Experimental results also show that, we can catch more than 90% of the reported errors.

We go over some issues about phonological similarity in Section 2, elaborate how we extend and apply Cangjie codes for simplified Chinese in Section 3, present details about our experiments and observations in Section 4, and discuss some technical issues in Section 5.

2 Phonologically Similar Characters

The pronunciation of a Chinese character involves a sound, which consists of the nucleus and an optional onset, and a tone. In Mandarin Chinese, there are four tones. (Some researchers include the fifth tone.)

In our work, we consider four categories of phonological similarity between two characters: same sound and same tone (**SS**), same sound and different tone (**SD**), similar sound and same tone (**MS**), and similar sound and different tone (**MD**).

We rely on the information provided in a lexicon (Dict, 2010) to determine whether two characters have the same sound or the same tone. The judgment of whether two characters have similar sound should consider the language experience of an individual. One who live in the southern and one who live in the northern China may have quite different perceptions of "similar" sound. In this work, we resort to the confusion sets observed in a psycholinguistic study conducted at the Academic Sinica.

Some Chinese characters are heteronyms. Let C_1 and C_2 be two characters that have multiple pronunciations. If C_1 and C_2 share one of their pronunciations, we consider that C_1 and C_2 belong to the SS category. This principle applies when we consider phonological similarity in other categories.

One challenge in defining similarity between characters is that the pronunciations of a character can depend on its context. The most common example of tone sandhi in Chinese (Chen, 2000) is that the first third-tone character in words formed by two adjacent third-tone characters will be pronounced in the second tone. At present, we ignore the influences of context when determining whether two characters are phonologically similar.

Although we have confined our definition of phonological similarity to the context of the Mandarin Chinese, it is important to note the influence of sublanguages within the Chinese language family will affect the perception of phonological similarity. Sublanguages used in different areas in China, e.g., Shanghai, Min, and Canton share the same written forms with the Mandarin Chinese, but have quite different though related pronunciation systems. Hence, people living in different areas in China may perceive phonological similarity in very different ways. The study in this direction is beyond the scope of the current study.

3 Visually Similar Characters

Figure 1 shows four groups of visually similar characters. Characters in group 1 and group 2 differ subtly at the stroke level. Characters in group 3 share the components on their right sides. The shared component of the characters in group 4 appears at different places within the characters.

Radicals are used in Chinese dictionaries to organize characters, so are useful for finding visually similar characters. The characters in group 1 and group 2 belong to the radicals "田" and "讠", respectively. Notice that, although the radical for group 2 is clear, the radical for group 1 is not obvious because "田" is not a standalone component.

However, the shared components might not be the radicals of characters. The shared components in groups 3 and 4 are not the radicals. In

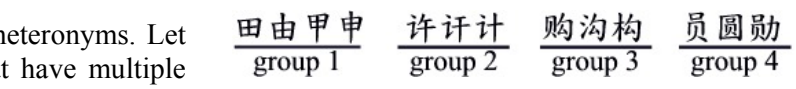

Figure 1. Examples of visually similar characters

many cases, radicals are semantic components of Chinese characters. In groups 3 and 4, the shared components carry information about the pronunciations of the characters. Hence, those characters are listed under different radicals, though they do look similar in some ways.

Hence, a mechanism other than just relying on information about characters in typical lexicons is necessary, and we will use the extended Cangjie codes for finding visually similar characters.

3.1 Cangjie Codes for Simplified Chinese

Table 1 shows the Cangjie codes for the 13 characters listed in Figure 1 and five other characters. The "ID" column shows the identification number for the characters, and we will refer to the i^{th} character by c_i, where i is the ID. The "CC" column shows the Chinese characters, and the "Cangjie" column shows the Cangjie codes. Each symbol in the Cangjie codes corresponds to a key on the keyboard, e.g. "田" and "中" collocate with "W" and "L", respectively. Information about the complete correspondence is available on the Wikipedia[2].

Using the Cangjie codes saves us from using image processing methods to determine the degrees of similarity between characters. Take the Cangjie codes for the characters in group 2 (c_5, c_6, and c_7) for example. It is possible to find that the characters share a common component, based on the shared substrings of the Cangjie codes, i.e., "戈女". Using the common substring (shown in black bold) of the Cangjie codes, we may also find the shared component "勾" for characters in group 3 (c_{10}, c_{11}, and c_{12}), the shared component "员" in c_{13} and c_{14}, the shared component "力" in c_{15} and c_{16}, and the shared component "孟" in c_{16} and c_{17}.

Despite the perceivable advantages, these original Cangjie codes are not good enough. In order to maintain efficiency in inputting Chinese characters, the Cangjie codes have been limited to no more than five keys. Thus, users of the Cangjie input method must familiarize themselves with the principles for simplifying the Cangjie codes. While the simplified codes help the input efficiency, they also introduce difficulties and ambiguities when we compare the Cangjie codes for computing similar characters. The prefix "弓一" in c_{16} and c_{17} can represent "孟", "鱼" (e.g., c_8), and "马" (e.g., c_9). Characters whose Cangjie codes include "弓一" may contain any of these three components, but they do not really look alike.

ID	CC	Cangjie	ID	CC	Cangjie
1	田	田	10	购	月人**心戈**
2	由	中田	11	沟	水**心戈**
3	甲	田中	12	构	木**心戈**
4	申	中田中	13	员	**口月人**
5	许	**戈女**人十	14	圆	田**口月人**
6	讦	**戈女**一十	15	勋	**口人大尸**
7	计	**戈女**十	16	劲	**弓一大尸**
8	鲳	**弓一**日日	17	颈	**弓一**一月人
9	驹	**弓一**心口	18	经	女一**弓人一**

Table 1. Examples of Cangjie codes

Therefore, we augment the original Cangjie codes by using the complete Cangjie codes and annotate each Chinese character with a layout identification that encodes the overall contours of the characters. This is how Liu and his colleagues (2008) did for the Cangjie codes for traditional Chinese characters, and we employ a similar exploration for the simplified Chinese.

3.2 Augmenting the Cangjie Codes

Figure 2 shows the twelve possible layouts that are considered for the Cangjie codes for simplified Chinese characters. Some of the layouts contain smaller areas, and the rectangles show a subarea within a character. The smaller areas are assigned IDs between one and three. Notice that, to maintain read-ability of the figures, not all IDs for subareas are shown in Figure 2. An example character is provided below each layout. From left to right and from top to bottom, each layout is assigned an identification number from 1 to 12. For example, the layout ID of "国" is 8. "国" has two parts, i.e., "口" and "玉".

Researchers have come up with other ways to

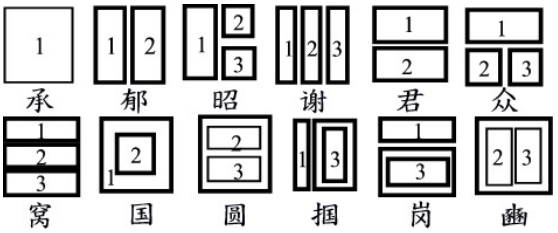

Figure 2. Layouts of Chinese characters

[2] en.wikipedia.org/wiki/Cangjie_input_method#Keyboard_layout ; last visited on 22 April 2010.

decompose individual Chinese characters. The Chinese Document Lab at the Academia Sinica proposed a system with 13 operators for describing the relationships among components in Chinese characters (CDL, 2010). Lee (2010b) propose more than 30 possible layouts.

The layout of a character affects how people perceive visual similarity between characters. For instance, c_{16} in Table 1 is more similar to c_{17} than to c_{18}, although they share "⺕". We rely on the expertise in Cangjie codes reported in (Lee, 2010a) to split the codes into parts.

Table 2 shows the extended codes for some characters listed in Table 1. The "ID" column provides links between the characters listed in both Table 1 and Table 2. The "CC" column shows the Chinese characters. The "LID" column shows the identifications for the layouts of the characters. The columns with headings "P1", "P2", and "P3" show the extended Cangjie codes, where "P*i*" shows the i^{th} part of the Cangjie codes, as indicated in Figure 2.

We decide the extended codes for the parts with the help of computer programs and subjective judgments. Starting from the original Cangjie codes, we can compute the most frequent substrings just like we can compute the frequencies of n-grams in corpora (cf. Jurafsky and Martin, 2009). Computing the most common substrings in the original codes is not a complex task because the longest original Cangjie codes contain just five symbols.

Often, the frequent substrings are simplified codes for popular components in Chinese characters, e.g., "讠" and "⺕". The original codes for "讠" and "⺕" are "戈弓女" and "弓人一", but they are often simplified to "戈女" and "弓一", respectively. When simplified, "讠" have the same Cangjie code with "戉", and "⺕" have the same Cangjie code with "马" and "鱼".

After finding the frequent substrings, we verify whether these frequent substrings are simplified codes for meaningful components. For meaningful components, we replace the simplified codes with complete codes. For instance the Cangjie codes for "许" and "讦" are extended to include "弓" in Table 2, where we indicate the extended keys that did not belong to the original Cangjie codes in boldface and with a surrounding box. Most of the non-meaningful frequent substrings have two keys: one is the last key of a part, and the other is the first key of another part. They were by observed by coincidence.

ID	CC	LID	P1	P2	P3
5	许	2	戈**弓**女	人十	
6	讦	2	戈**弓**女	一十	
7	计	2	戈**弓**女	十	
10	购	10	月人	心	戈
11	沟	10	水	心	戈
12	构	10	木	心	戈
13	员	5	口	月人	
14	圆	9	田	口	月人
15	勋	2	口**月**人	大尸	
16	劲	2	弓**人**一	大尸	
17	颈	2	弓**人**一	一月人	
18	经	3	女**女**一	弓人	一
19	恸	4	心	一**一**戈	**大**尸

Table 2. Examples of extended Cangjie codes

Although most of the examples provided in Table 2 indicate that we expand only the first part of the Cangjie codes, it is absolutely possible that the other parts, i.e., P2 and P3, may need to be extended too. c_{19} shows such an example.

Replacing simplified codes with complete codes not only help us avoid incorrect matches but also help us find matches that would be missed due to simplification of Cangjie codes. Using just the original Cangjie codes in Table 1, it is not easy to determine that c_{18} ("经") in Table 1 shares a component ("⺕") with c_{16} and c_{17} ("劲" and "颈"). In contrast, there is a chance to find the similarity with the extended Cangjie codes in Table 2, given that all of the three Cangjie codes include "弓人一".

We can see an application of the LIDs, using "劲", "颈" and "经" as an example. Consider the case that we want to determine which of "颈" and "经" is more similar to "劲". Their extended Cangjie codes will indicate that "颈" is the answer to this question for two reasons. First, "劲" and "颈" belong to the same type of layout; and, second, the shared components reside at the same area in "劲" and "颈".

3.3 Similarity Measures

The main differences between the original and the extended Cangjie codes are the degrees of details about the structures of the Chinese characters. By recovering the details that were ignored in the original codes, our programs will be

better equipped to find the similarity between characters.

In the current study, we experiment with three different scoring methods to measure the visual similarity between two characters based on their extended Cangjie codes. Two of these methods had been tried by Liu and his colleagues' study for traditional Chinese characters (Liu et al., 2009b-c). The first method, denoted **SC1**, considers the total number of matched keys in the matched parts (without considering their part IDs). Let c_i denote the i^{th} character listed in Table 2. We have SC1(c_{15}, c_{16}) = 2 because of the matched "大尸". Analogously, we have SC1(c_{19}, c_{16}) = 2.

The second method, denoted **SC2**, includes the score of SC1 and considers the following conditions: (1) add one point if the matched parts locate at the same place in the characters and (2) if the first condition is met, an extra point will be added if the characters belong to the same layout. Hence, we have SC2(c_{15}, c_{16}) =SC1(c_{15}, c_{16})+1+1=4 because (1) the matched "大尸" locate at P2 in both characters and (2) c_{15} and c_{16} belong to the same layout. Assuming that c_{16} belongs to layout 5, than SC2(c_{15}, c_{16}) would become 3. In contrast, we have SC2(c_{19}, c_{16})=2. No extra weights for the matching "大尸" because it locates at different parts in the characters. The extra weight considers the spatial influences of the matched parts on the perception of similarity.

While splitting the extended Cangjie codes into parts allows us to tell that c_{15} is more similar to c_{16} than to c_{19}, it also creates a new barrier in computing similarity scores. An example of this problem is that SC2(c_{17}, c_{18})=0. This is because that "弓人一" at P1 in c_{17} can match neither "弓人" at P2 nor "一" at P3 in c_{18}.

To alleviate this problem, we consider **SC3** which computes the similarity in three steps. First, we concatenate the parts of a Cangjie code for a character. Then, we compute the longest common subsequence (**LCS**) (cf. Cormen et al., 2009) of the concatenated codes of the two characters being compared, and compute a Dice's coefficient (cf. Croft et al., 2010) as the similarity. Let X and Y denote the concatenated, extended Cangjie codes for two characters, and let Z be the LCS of X and Y. The similarity is defined by the following equation.

$$Dice_{LCS} = \frac{2 \times |Z|}{|X| + |Y|}, \text{ where } |S| \text{ is the length of string } S \quad (1)$$

We compute another Dice's coefficient between X and Y. The formula is the similar to (1), except that we set Z to the longest common *consecutive* subsequence. We call this score $Dice_{LCCS}$. Notice that $Dice_{LCCS} \le Dice_{LCS}$, $Dice_{LCCS} \le 1$, and $Dice_{LCS} \le 1$. Finally, SC3 of two characters is the sum of their SC2, $10 \times Dice_{LCCS}$, and $5 \times Dice_{LCS}$. We multiply the Dice's coefficients with constants to make them as influential as the SC2 component in SC3. The constants were not scientifically chosen, but were selected heuristically.

4 Error Analysis and Evaluation

We evaluate the effectiveness of using the phonologically and visually similar characters to captures errors in simplified Chinese words with two lists of reported errors that were collected from the Internet.

4.1 Data Sources

We need two types of data for the experiments. The information about the pronunciation and structures of the Chinese characters help us generate lists of similar characters. We also need reported errors so that we can evaluate whether the similar characters catch the reported errors.

A lexicon that provides the pronunciation information about Chinese characters and a database that contains the extended Cangjie codes are necessary for our programs to generate lists of characters that are phonologically and visually similar to a given character.

It is not difficult to acquire lexicons that show standard pronunciations for Chinese characters. As we stated in Section 2, the main problem is that it is not easy to predict how people in different areas in China actually pronounce the characters. Hence, we can only rely on the standards that are recorded in lexicons.

With the procedure reported in Section 3.2, we built a database of extended Cangjie codes for the simplified Chinese. The database was designed to contain 5401 common characters in the BIG5 encoding, which was originally designed for the traditional Chinese. After converting the traditional Chinese characters to the simplified counterparts, the database contained only 5170

different characters.

We searched the Internet for reported errors that were collected in real-world scenarios, and obtained two lists of errors. The first list[3] came from the entrance examinations for senior high schools in China, and the second list[4] contained errors observed at senior high schools in China. We used 160 and 524 errors from the first and the second lists, respectively, and we refer to the combined list as the **Ilist**. An item of reported error contained two parts: the correct word and the mistaken character, both of which will be used in our experiments.

4.2 Preliminary Data Analysis

Since our programs can compare the similarity only between characters that are included in our lexicon, we have to exclude some reported errors from the Ilist. As a result, we used only 621 errors in this section.

Two native speakers subjectively classified the causes of these errors into three categories based on whether the errors were related to phonological similarity, visual similarity, or neither. Since the annotators did not always agree on their classifications, the final results have five interesting categories: "P", "V", "N", "D", and "B" in Table 3. P and V indicate that the annotators agreed on the types of errors to be related to phonological and visual similarity, respectively. N indicates that the annotators believed that the errors were not due to phonological or visual similarity. D indicates that the annotators believed that the errors were due to phonological or visual similarity, but they did not have a consensus. B indicates the intersection of P and V.

Table 3 shows the percentages of errors in these categories. To get 100% from the table, we can add up P, V, N, and D, and subtract B from the total. In reality there are errors of type N, and Liu and his colleagues (2009b) reported this type of errors. Errors in this category happened to be missing in the Ilist. Based on our and Liu's observations, the percentages of phonological and visual similarities contribute to the errors in simplified and traditional Chinese words with similar percentages.

	P	V	N	D	B
Ilist	83.1	48.3	0	3.7	35.1

Table 3. Percentages of types of errors

[3] www.0668edu.com/soft/4/12/95/2008/2008091357140.htm ; last visited on 22 April 2010.

[4] gaozhong.kt5u.com/soft/2/38018.html; last visited on 22 April 2010.

4.3 Experimental Procedure

We design and employ the `ICCEval` procedure for the evaluation task.

At step 1, given the correct word and the correct character to be intentionally replaced with incorrect characters, we created a list of characters based on the selection criterion. We may choose to evaluate phonologically or visually similar characters. For a given character, `ICCEval` can generate characters that are in the SS, SD, MS, and MD categories for phonologically similar characters (cf. Section 2). For visually similar characters, `ICCEval` can select characters based on SC1, SC2, and SC3 (cf. Section 3.3). In addition, `ICCEval` can generate a list of characters that belong to the same radical and have the same number of strokes with the correct character. In the experimental results, we refer to this type of similar characters as **RS**.

At step 2, for a correct word that people originally wanted to write, we replaced the correct character with an incorrect character with the characters that were generated at step 1, submitted the incorrect word to Google AJAX Search

```
Procedure ICCEval
Input:
  ccr: the correct character; cwd:
  the correct word; crit: the selec-
  tion criterion; num: number of re-
  quested characters; rnk: the cri-
  terion to rank the incorrect
  words;
Output: a list of ranked candidates
    for ccr
Steps:
1.  Generate a list, L, of charac-
    ters for ccr with the specified
    criterion, crit. When using SC1,
    SC2, or SC3 to select visually
    similar characters, at most num
    characters will be selected.
2.  For each c in L, replace ccr in
    cwd with c, submit the resulting
    incorrect word to Google, and
    record the ENOP.
3.  Rank the list of incorrect words
    generated at step 2, using the
    criterion specified by rnk.
4.  Return the ranked list.
```

API, and extracted the estimated numbers of pages (ENOP) [5] that contained the incorrect words. In an ordinary interaction with Google, an ENOP can be retrieved from the search results, and it typically follows the string "`Results 1-10 of about`" on the upper part of the browser window. Using the AJAX API, we just have to parse the returned results with a simple method.

Larger ENOPs for incorrect words suggest that these words are incorrect words that people frequently used on their web pages. Hence, we ranked the similar characters based on their ENOPs at step 3, and return the list.

Since the reported errors contained information about the incorrect ways to write the correct words, we could check whether the real incorrect characters were among the similar characters that our programs generated at step 1 (inclusion tests). We could also check whether the actual incorrect characters were ranked higher in the ranked lists (ranking tests).

Take the word "和藹可亲" as an example. In the collected data, it is reported that people wrote this word as "和靄可亲", i.e., the second character was incorrect. Hoping to capture the error, `ICCEval` generated a list of possible substitutions for "藹" at step 1. Depending on the categories of sources of errors, `ICCEval` generated a list of characters. When aiming to test the effectiveness of visually similar characters, we could ask `ICCEval` to apply SC3 to generate a list of alternatives for "藹", possibly including "靄", "谒", "葛", and other candidates. At step 2, we created and submitted query strings "和靄可亲", "和谒可亲", and "和葛可亲" to obtain the ENOPs for the candidates. If the ENOPs were, respectively, 410000, 26100, and 7940, these candidates would be returned in the order of "靄", "谒", and "葛". As a result, the returned list contained the actual incorrect character "靄", and placed "靄" on top of the ranked list.

Notice that we considered the contexts in which the incorrect characters appeared to rank. We did not rank the incorrect characters with just the unigrams. In addition, although this running example shows that we ranked the characters directly with the ENOPs, we also ranked the list of alternatives with pointwise mutual information:

$$PMI(C, X) = \frac{\Pr(C \wedge X)}{\Pr(C) \times \Pr(X)}, \quad (2)$$

where X is the candidate character to replace the correct character and C is the correct word excluding the correct character to be replaced. To compute the score of replacing "藹" with "靄" in "和藹可亲", X = "靄", C="和□可亲", and ($C \wedge X$) is "和靄可亲". (□ denotes a character to be replaced.) PMI is a common tool for judging collocations in natural language processing. (cf. Jurafsky and Martin, 2009).

It would demand very much computation effort to find Pr(C). Fortunately, we do not have to consider Pr(C) because it is a common denominator for all incorrect characters. Let X_1 and X_2 be two competing candidates for the correct character. We can ignore Pr(C) because of the following relationship.

$$PMI(C, X_1) \geq PMI(C, X_2) \Leftrightarrow \frac{\Pr(C \wedge X_1)}{\Pr(X_1)} \geq \frac{\Pr(C \wedge X_2)}{\Pr(X_2)}$$

Hence, X_1 prevails if $score(C, X_1)$ is larger.

$$score(C, X) = \frac{\Pr(C \wedge X)}{\Pr(X)} \quad (3)$$

In our work, we approximate the probabilities used in (3) by the corresponding frequencies that we can collect through Google, similar to the methods that we used to collect the ENOPs.

4.4 Experimental Results: Inclusion Tests

We ran `ICCEval` with 621 errors in the Ilist. The experiments were conducted for all categories of phonological and visual similarity. When using SS, SD, MS, MD, and RS as the selection criterion, we did not limit the number of candidate characters. When using SC1, SC2, and SC3 as the criterion, we limited the number candidates to be no more than 30. We consider only words that the native speakers have consensus over the causes of errors. Hence, we dropped those 3.7% of words in Table 3, and had just 598 errors. The ENOPs were obtained during March and April 2010.

Table 4 shows the chances that the lists, gen-

	SS	SD	MS	MD	Phone
Ilist	82.6	29.3	1.7	1.6	97.3
	SC1	SC2	SC3	RS	Visual
Ilist	78.3	71.0	87.7	1.3	90.0

Table 4. Chances of the recommended list contains the incorrect character

[5] According to (Croft et al., 2010), the ENOPs may not reflect the actual number of pages on the Internet.

erated with different `crit` at step 1, contained the incorrect character in the reported errors. In the Ilist, there were 516 and 300[6] errors that were related to phonological and visual similarity, respectively. Using the characters generated with the SS criterion, we captured 426 out of 516 phone-related errors, so we showed 426/516 = 82.6% in the table.

Results in Table 4 show that we captured phone-related errors more effectively than visually-similar errors. With a simple method, we can compute the union of the characters that were generated with the SS, SD, MS, and MD criteria. This integrated list suggested how well we captured the errors that were related to phones, and we show its effectiveness under "Phone". Similarly, we integrated the lists generated by SC1, SC2, SC3, and RS to explore the effectiveness of finding errors that are related to visual similarity, and the result is shown under "Visual".

4.5 Experimental Results: Ranking Tests

To put the generated characters into work, we wish to put the actual incorrect character high in the ranked list. This will help the efficiency in supporting computer assisted test-item writing. Having short lists that contain relatively more confusing characters may facilitate the data preparation for psycholinguistic studies.

At step 3, we ranked the candidate characters by forming incorrect words with other characters in the correct words as the context and submitted the words to Google for ENOPs. The results of ranking, shown in Table 5, indicate that we may just offer the leading five candidates to cover the actual incorrect characters in almost all cases.

The "Total" column shows the total number of errors that were captured by the selection criterion. The column "R*i*" shows the percentage of all errors, due to phonological or visual similarity, that were re-created and ranked i^{th} at step 3 in `ICCEVAL`. The row headings show the selection criteria that were used in the experiments. For instance, using SS as the criterion, 70.3% of actual phone-related errors were rank first, 7.4% of the phone-related errors were ranked second, etc. If we recommended only 5 leading incorrect characters only with SS, we would have captured the actual incorrect characters that were phone related 81.6% (the sum of R1 to R5) of the time. For errors that were related to visual similarity, recommending the top five candidates with SC3 would capture the actual incorrect characters 87.1% of the time. Since we do not show the complete distributions, the sums over the rows are not 100%. In the current experiments, the worst rank was 21.

	Total	R1	R2	R3	R4	R5
SS	426	70.3	7.4	2.9	0.4	0.6
SD	151	25.6	2.7	0.6	0.0	0.4
MS	9	1.4	0.4	0.0	0.0	0.0
MD	8	1.6	0.0	0.0	0.0	0.0
SC1	235	61.3	10.3	4.3	2.0	0.3
SC2	213	53.7	11.0	3.7	2.3	0.3
SC3	263	66.7	12.7	5.7	1.7	0.3
RS	4	1.3	0.0	0.0	0.0	0.0

Table 5. Ranking the candidates

We also used PMI to rank the incorrect words. Due to page limits, we cannot show complete details about the results. The observed distributions in ranks were not very different from those shown in Table 5.

5 Discussion

Compared with Liu et al.'s analysis (2009b-c) for the traditional Chinese, the proportions of errors related to phonological factors are almost the same, both at about 80%. The proportion of errors related to visual factors varied, but the averages in both studies were about 48%. A larger scale of study is needed for how traditional and simplified characters affect the distributions of errors. Results shown in Table 4 suggest that it is relatively easy to capture errors related to visual factors in simplified Chinese. Although we cannot elaborate, we note that Cangjie codes are not good for comparing characters that have few strokes, e.g., c_1 to c_4 in Table 1. In these cases, the coding method for Wubihua input method (Wubihua, 2010) should be applied.

Acknowledgement

This research was supported in part by the research contract NSC-97-2221-E-004-007-MY2 from the National Science Council of Taiwan. We thank the anonymous reviewers for constructive comments. Although we are not able to respond to all the comments

[6]The sum of 516 and 300 is larger than 598 because some of the characters are similar both phonologically and visually.

in this paper, we have done so in an extended version of this paper.

References

Cangjie. 2010. Last visited on 22 April 2010: en.wikipedia.org/wiki/Cangjie_input_method.

CDL. 2010. Chinese document laboratory, Academia Sinica. Last visited on 22 April, 2010; cdp.sinica.edu.tw/cdphanzi/. (in Chinese)

Chen, Matthew. Y. 2000. *Tone Sandhi: Patterns across Chinese Dialects*, (Cambridge Studies in Linguistics 92). Cambridge University Press.

Chu, Bong-Foo. 2010. *Handbook of the Fifth Generation of the Cangjie Input Method*. last visited on 22 April 2010: www.cbflabs.com/book/5cjbook/. (in Chinese)

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms*, third edition. MIT Press.

Croft, W. Bruce, Donald Metzler, and Trevor Strohman, 2010. *Search Engines: Information Retrieval in Practice*, Pearson.

Dict. 2010. Last visited on 22 April 2010, www.cns11643.gov.tw/AIDB/welcome.do

Fan, Kuo-Chin, Chang-Keng Lin, and Kuo-Sen Chou. 1995. Confusion set recognition of on-line Chinese characters by artificial intelligence technique. *Pattern Recognition*, **28**(3):303−313.

HanDict. 2010. Last visit on 22 April 2010, www.zdic.net/appendix/f19.htm.

Jurafsky, Daniel and James H. Martin. 2009. *Speech and Language Processing*, second edition, Pearson.

Kuo, Wen-Jui, Tzu-Chen Yeh, Jun-Ren Lee, Li-Fen Chen, Po-Lei Lee, Shyan-Shiou Chen, Low-Tone Ho, Daisy L. Hung, Ovid J.-L. Tzeng, and Jen-Chuen Hsieh. 2004. Orthographic and phonological processing of Chinese characters: An fMRI study. *NeuroImage*, **21**(4):1721−1731.

Lee, Chia-Ying, Jie-Li Tsai, Hsu-Wen Huang, Daisy L. Hung, Ovid J.-L. Tzeng. 2006. The temporal signatures of semantic and phonological activations for Chinese sublexical processing: An even-related potential study. *Brain Research*, **1121**(1):150-159.

Lee, Hsiang. 2010a. *Cangjie Input Methods in 30 Days 2*. Foruto. Last visited on 22 April 2010: input.foruto.com/cccls/cjzd.html.

Lee, Mu. 2010b. A quantitative study of the formation of Chinese characters. Last visited on 22 April 2010: chinese.exponode.com/0_1.htm. (in Chinese)

Liu, Chao-Lin, and Jen-Hsiang Lin. 2008. Using structural information for identifying similar Chinese characters. *Proc. of the 46th Annual Meeting of the Association for Computational Linguistics*, short papers, 93−96.

Liu, Chao-Lin, Kan-Wen Tien, Yi-Hsuan Chuang, Chih-Bin Huang, and Juei-Yu Weng. 2009a. Two applications of lexical information to computer-assisted item authoring for elementary Chinese. *Proc. of the 22nd Int'l Conf. on Industrial Engineering & Other Applications of Applied Intelligent Systems*, 470−480.

Liu, Chao-Lin, Kan-Wen Tien, Min-Hua Lai, Yi-Hsuan Chuang, and Shih-Hung Wu. 2009b. Capturing errors in written Chinese words. *Proc. of the 47th Annual Meeting of the Association for Computational Linguistics*, short papers, 25−28.

Liu, Chao-Lin, Kan-Wen Tien, Min-Hua Lai, Yi-Hsuan Chuang, and Shih-Hung Wu. 2009c. Phonological and logographic influences on errors in written Chinese words. *Proc. of the 7th Workshop on Asian Language Resources*, the 47th Annual Meeting of the ACL, 84−91.

Liu, Cheng-Lin, Stefan Jaeger, and Masaki Nakagawa. 2004. Online recognition of Chinese characters: The state-of-the-art. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, **26**(2):198−213.

Wubihua. 2010. Last visited on 22 April 2010: en.wikipedia.org/wiki/Wubihua_method.

Yeh, Su-Ling, and Jing-Ling Li. 2002. Role of structure and component in judgments of visual similarity of Chinese Characters. *Journal of Experimental Psychology: Human Perception and Performance*, **28**(4):933−947.

KONG, S.C., et al. (Eds.), ICCE2009; ©2009 Asia-Pacific Society for Computers in Education.

Computer Assisted Creation of Items for Scrambled Sentence Tests

Chao-Lin Liu Chih-Bin Huang Ying-Tse Sun Wei-Ti Kuo
Department of Computer Science, National Chengchi University, Taiwan
{chaolin,96753014,93703038,94703041}@nccu.edu.tw

Abstract. We apply techniques of natural language processing to support the creation of special scrambled sentences that allow only specific word orders. The scrambled sentences are useful for students to practice their knowledge about grammars. It takes two steps to create a test item for scrambled-sentence tests. We create a set of grammatical alternative sentences from the target sentence, and make sure that students will rebuild the target sentence by pegging some of the words in the target sentence. The proposed methods can automatically and effectively peg words to single out a specific sentence from a set of sentences. We also employ the Stanford parser and propose a practical heuristic principle to help teachers exclude a potentially large number of alternative grammatical orderings of a set of words in the scrambled sentence.

Keywords: scrambled sentence tests, computer assisted item generation, grammar learning, natural language processing

1 Introduction

Techniques that were originally designed for natural language processing (NLP) have proved to be instrumental for applications for computer assisted language learning (CALL) [1, 3, 7]. In this paper, we report an application of NLP techniques for grammar learning.

Word orders are important in conveying the correct meaning in almost all languages, so learning the correct word orders are crucial even for beginning learners of languages (cf. some examples at http://www.manythings.org/ss/ and http://www.msrossbec.com/scrambleintro.shtml). Placing students in actual conversations is common for students to practice knowledge of word orders. However, some students might not be ready for such challenging field tests, so a less stressful environment is necessary for those who are not completely ready for direct conversations.

Constructing sentences from scrambled sentences offers an alternative for students of intermediate competence. In these practices, sentences are segmented and scrambled to create a set of words or phrases, and students have to reconstruct the original sentences with the given text segments. Taking advantage of the information contained in parse trees, Liu et al. [5] segmented the sentences into different numbers of pieces to make the test items adaptive to students' competence levels.

A sentence can be segmented at coarse or fine levels, and a student may find two or more grammatical orders of the resulting segments. When a teacher prefers to avoiding multiple answers to a test item, just segmenting the sentences based on parse trees become insufficient, and more techniques are in need.

The safest way to make sure that there is only way to build a sentence from a set of words is to rule out all but the target sentence by pegging some words. Given a set of sentences, it is not very difficult to find such words to be anchored. Our experience show that the task of generating all grammatical permutations of a set of words turns out to be more challenging that it appears [1, 4].

In Section 2, we present more background information about the work on scrambling sentences for grammar learners. In Section 3, we propose methods to select and peg some words for a set of sentences to achieve a unique ordering. In Section 4, we attempt to employ the Stanford parser (http://nlp.stanford.edu:8080/parser/) to find all grammatical arrangements of a set of words and phrases. With such a capability, we will reduce the burden of teachers. In Section 5, we report the results when we repeated the work reported in the previous section with a categorical parser. In Section 6, we propose a practical heuristic to solve the discussed problems.

2 Problem Definition

We employ parsing techniques to split a given sentence. Figure 1 shows a parse tree, produced by the Stanford parser, for "The new bike is better than that old car". This sentence can be split in multiple ways, e.g., {"this new bike", "is better than", "that old car"} or {"this", "new", "bike", "is better than", "that", "old", "car"}. An instance of sentence reconstruction practice is to provide the segments {"is better than", "this new bike", "that old car"} to students and expect them to come up with the original sentence.

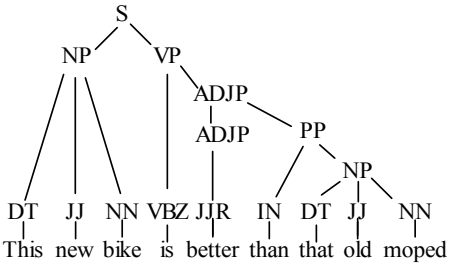

Figure 1. A sample parse tree

Splitting the given sentence at different granularities offers a way to control the difficulty of the test items, and is a useful strategy in adaptive assessment. Rebuilding the sentence from {"is better than", "this new bike", "that old car"} is easier than rebuilding the sentence from {"this", "new", "bike", "is better than", "that", "old", "car"}. Splitting the sentence into more segments, all else being equal, generally makes the task of reconstruction of the sentence more challenging. Hitting the correct order from 3!(=6) arrangements by chance is more likely to occur than hitting the correct order from 7!(=5040) arrangements by chance.

There might be different ways to rebuild a correct sentence for a given segmentation of the original sentence. When we segment the original sentence into more segments, we may have allowed more possible answers. For instance, we may build at least 48 sentences from {"this", "new", "bike", "is", "better", "than", "that", "old", "car"} (cf. Table 1 on the next page).

When we would like the students to practice a particular sentence pattern, we may want to force the students to rebuild the sentence in a specific manner, e.g., assertive sentences vs. yes/no questions. One may rely on the contextual information to exclude some of the syntactically acceptable arrangements as Liu et al. [5] suggested. If the sentence in Figure 1 was preceded by an utterance that involved a new bike in a conversation, then "this old bike" would become less plausible in the expected ordering.

An automatic method to fix some of the words at their original locations makes certain that there is exactly one acceptable solution. If we split the sentence in Figure 1 into nine words, there will be at least 48 grammatical arrangements. If we fix "new", "that" and "car" at the second, the seventh and the ninth words in the sentence, respectively, there will be only way rebuild the sentence.

In addition, when preparing a test item, a teacher typically provides only one sentence—The one the teachers would like the students to rebuild. To facilitate the preparation of test items, we attempt to find all of the alternative arrangements of the words that constitute the given sentence. For the sentence used in Figure 1, we would like to create sentences like "This new car is better than that old bike" and "Is this new bike better than that old bike" for the teachers, and help the teachers rule out these grammatical alternatives by pegging some selected words.

3 Pegging Words for a Unique Order

Assume that we have a way to find all grammatical arrangements of the words for a sentence like the one shown in Figure 1. We present algorithms to select and fix the words that will single out the desirable arrangements. It is very likely that, for educational purposes, a teacher may want to fix certain words of her/his choices. The user interface of our system certainly can accommodate this need, but we will not consider this need when discussing the design of the internal algorithm.

Assume that there are m tokens, w_1, w_2, …, w_i, …, w_m, in a given sentence, s_g, and that we have a set of n sentences, $S=\{s_1, s_2, \ldots, s_j, \ldots, s_n\}$, that are formed with those m tokens in s_g. Hence, we have $s_g \in S$. Figure 2 shows the algorithm that will find the minimum number of words to peg and to allow s_g to be the answer to the sentence reconstruction task. In the algorithm, we identify the sentences that share the same tokens at the same positions as s_g. The sentences that meet such criteria are called "surviving" sentences. The algorithm aims at reducing the number of surviving sentences by pegging more and more tokens. By construction, the algorithm will iterate at most (m-1) times, when it chooses to peg (m-1) tokens. In addition, when it happens to peg (m-1) tokens, s_g will be the last sentence in S at Step 5.

We enumerated 48 grammatical arrangements of the words in the sentence in Figure 1, and we list the sentences in Table 1. We ran the Word-Pegging algorithm with these

Algorithm: Word-Pegging
Input: $S = \{s_1, s_2, \ldots, s_j, \ldots, s_n\}$ and
$s_g = w_1 w_2 \ldots w_i \ldots w_m$; where $s_g \in S$
Output: the set of indexes of tokens to be pegged
Steps:

1. Set W to $\varnothing$
2. If $k \notin W$, set c_k to the number of sentences in S whose k-th token is the same as w_k. If $k \in W$, set c_k to $(n+1)$
3. Find the index v in $[1,m]$, such that c_v is the smallest among all c_u for any $u \neq v$ (more discussion about the choice of v in the paper)
4. Add v to W. Remove from S the sentences whose v-th token is not the same as w_v
5. If there is only one sentence in S, return W; otherwise, return to Step 2

Figure 2. An algorithm for pegging words

48 arrangements. The algorithm returned the indexes for "the", "new", and "bike". In fact, this is not the only choice, and pegging words in {"new", "bike", "that"} will also make the original sentence the only answer among the 48 arrangements. If the original sentence was "Is this new bike better than that old car", Word-Pegging will recommend {"is", "this", "new", "bike"}. If we modify and allow Word-Pegging to find alternative ways of pegging the words, {"this", "bike", "better", "old"} can be such an alternative.

A conceivable method to offer flexibility to teachers is that we modify Word-Pegging to compute all possible ways of pegging the words and provide these choices to the teacher. For instance, allowing the teachers to choose whether we should peg words in {"this", "new", "bike"}, {"new", "bike", "that"}, or {"new", "that", "car"}. This will allow the teachers to make the resulting test item fit specific teaching needs. However, there can be many such choices.

A more practical alternative to make Word-Pegging more flexible is to allow the teachers to specify a set of words that they want to peg. However, we do not include the algorithm due to page limits.

4 Seeking Admissible Arrangements

Recommending the tokens to peg to achieve unique answers resolves the first problem discussed in Section 2. The algorithms we presented and discussed in Section 3 will be sufficient when it is possible for the teachers to provide an exhaustive list of grammatical arrangements like those listed in Table 1. However it is not always easy for a human teacher to enumerate such grammatical permutations and provide the set of *S* as input to our algorithms. It will be more convenient for the teachers if our system can help them enumerate all possible arrangements of the words of the sentence provided by the teacher.

Table 1. Alternative sentences with ranks

48 Arrangements (rank)
this new bike is better than that old car (16)
that new bike is better than this old car (8)
this old bike is better than that new car (14)
this new car is better than that old bike (15)
that old bike is better than this new car (6)
this old car is better than that new bike (13)
that new car is better than this old bike (7)
that old car is better than this new bike (5)
is this new bike better than that old car (1181)
is that new bike better than this old car (1177)
is this old bike better than that new car (1179)
is this new car better than that old bike (1180)
is that old bike better than this new car (1175)
is this old car better than that new bike (1178)
is that new car better than this old bike (1176)
is that old car better than this new bike (1174)
this bike is better than that new old car (38)
this bike is better than that old new car (37)
that bike is better than this old new car (19)
that bike is better than this new old car (20)
this car is better than that new old bike (36)
this car is better than that old new bike (35)
that car is better than this new old bike (18)
that car is better than this old new bike (17)
this new old bike is better than that car (94)
this old new bike is better than that car (92)
that old new bike is better than this car (59)
that new old bike is better than this car (60)
this new old car is better than that bike (93)
this old new car is better than that bike (91)
that new old car is better than this bike (58)
that old new car is better than this bike (57)
is this bike better than that new old car (839)
is this bike better than that old new car (838)
is that bike better than this old new car (833)
is that bike better than this new old car (834)
is this car better than that new old bike (836)
is this car better than that old new bike (828)
is that car better than this new old bike (831)
is that car better than this old new bike (825)
is this new old bike better than that car (840)
is this old new bike better than that car (837)
is that old new bike better than this car (832)
is that new old bike better than this car (835)
is this new old car better than that bike (830)
is this old new car better than that bike (829)
is that new old car better than this bike (827)
is that old new car better than this bike (826)

An intuitive approach for generating the grammatical enumerations for a given sentence that contains *m* tokens is to compute all *m*! permutations and check their grammaticality. Doing so requires a categorical parser, which we do not have at this moment. Instead we employ the Stanford parser to parse the permutations.

A more challenging barrier is that parsing all of the *m*! permutations is feasible only for relatively small values of *m* because of computational costs. It took about 5.5 hours to parse the 9! (=362880) permutations of the sentence in Figure 1, when we ran the Stanford parser (version 2008-10-26) on a machine with 3.0G INTEL Core 2 Duo E8400 CPU and 2.0 G RAM running Windows XP SP3.

In fact, we did not receive categorical decisions for the grammaticality of the permutations from the Stanford parser. It would accept all of the permutations and provided the probabilities of the permutations being grammatically acceptable.

In general, these estimated probabilities are not good for comparing the acceptability of sentences of different lengths, but they serve as a good indication for sentences of the same length. In our case, we have an additional advantage that we are comparing the acceptability of the permutations of the same set of tokens.

Hence, we submitted the permutations to the Stanford parser, collected the probabilities of these permutations, and ranked them accordingly. Table 1 shows the ranks of the 48 legal arrangements in the 8! (=40320) permutations. We created the 40320 permutations by treating "more than" as a token in the sentence in Figure 1. The permutation "That is better than this old new car bike" received the highest probability in all permutations, where a noun is functioning as an adjective. Nevertheless, we did not treat it as an acceptable candidate in Table 1.

Given the ranking information, our job is to choose the α permutations that have the leading probabilities among all of these 40320 permutations as the input to the Word-Pegging algorithm. If we choose the top 100 arrangements, and run the Word-Pegging algorithm, the algorithm will recommend us to

peg the words in {"new", "bike", "that"}. This choice turns out to be correct selections for the sentence in Figure 1 and sentences in Table 1. Another possible choice is {"new", "that", "car"}. If we choose the top 400 arrangements, the algorithm will recommend us to peg words in {"new", "bike", "is", "that"}. This time, we place an unnecessary constraint to force the students to rebuild the sentence in Figure 1. Notice that the sentence in Figure 1 was ranked 16 as shown in Table 1. Furthermore, an inspection of the permutations shows that the permutations which received the top 10 probabilities all started the sentences with "That". Consequently, the Word-Pegging algorithm will find that we just have to peg the first word of the original sentence, i.e., "This", to force the students to arrange the words exactly the same as the sentence in Figure 1. This is certainly incorrect as none of the 10 leading permutations belonged to the grammatical set that we listed in Table 1.

The ranking information listed in Table 1 and the aforementioned examples show that a precise selection of the grammatical arrangements is not an easy task. The 48 permutations were ranked within about four dispersed intervals in the list. The minimum of these ranks is 5, the average is about 493, and the maximum is 1181. Hence, it will not be easy to find a convincing threshold, say α, to find α leading permutations to compute the pegging positions.

When we choose a large α, we will be forced to peg more tokens than necessary to exclude all of the (α-1) permutations. This is the case if we choose 400 leading permutations. At one extreme, if we consider too many superfluous permutations, we may have to peg (m-1) tokens for a sentence that has only m tokens, making the resulting test item useless. On the other hand, when we choose a very small α, we probably will miss those admissible arrangements, allowing multiple answers to the test items. This is the case if we choose 10 leading permutations.

5 Filtering with a Categorical Parser

A parser that can categorically judge the acceptability of a sentence will relieve us of the task of selecting the value of α in the previous section. It is easy to implement a chart parser (a.k.a. CYK parser; [6]) for this purpose. However, it is not trivial to acquire a set of grammar rules for the parser.

We relied on the Stanford parser to alleviate the problem. We submitted 7361 real sentences (48373 words in total) that we collected for learners of English to the Stanford parser, and extracted the rules from the parse trees. Since Stanford parser returned the most likely parse tree for a given sentence, we considered only one parse tree for one sentence at this stage. We obtained 985 rules. The most frequent rule occurred 4450 times, and 377 and 143 rules occurred only once and twice, respectively. The median frequency of these rules is 2 (the second point, from the left, in Figure 3). The average frequency is 42.07 (2.6 on the horizontal axis in Figure 3). Figure 3 shows the distribution of the frequencies of the rules. To make the rules work with our chart parser, we had to rewrite the rules into the Chomsky Normal Form (cf. [6]). Since the purpose of parsing the sentences with the chart parser was to verify grammaticality, it was fine with us to use such weakly equivalent grammars.

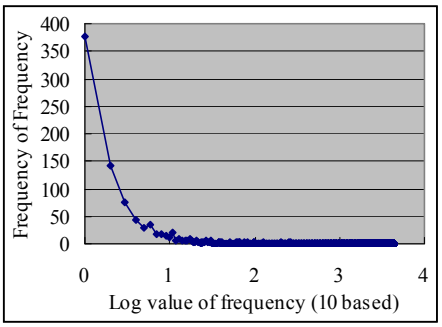

Figure 3. Frequency of frequencies of rules

Table 2 shows the nine sets of sentences that we used to evaluate the approach of using a categorical parser to find grammatical arrangements of words. The words in the first sentence in every set are separated by underscores. The underscores show how we segmented the original sentence into k chunks, and create all of the k! permutations of the chunks as strings that were then checked by the categorical parser. (Notice that we had segmented the sentences into different numbers of chunks.)

Take the first set of sentence for example. We segmented the original sentence "She can neither sing well nor dance beautifully" into {"she", "can", "neither", "sing well", "nor", "dance beautifully"}, and created all of the 6! permutations of these six chunks. These 6! permutations were checked by the chart parser for grammaticality. There were at least two grammatical sentences in each set of sentences in Table 2. The grammatical sentences were numbered for identification in both Table 2 and Table 3.

Table 2. More sentences used in the experiments (not all experimental results are included)

ID	Sentences
1.1	She_can_neither_sing well_nor_dance beautifully
1.2	She can neither dance beautifully nor sing well
2.1	They_are_sometimes_late_for_work
2.2	Sometimes they are late for work
2.3	They are late for work sometimes
2.4	They sometimes are late for work
6.1	One_of_my favorite hobbies_is_reading
6.2	Reading is one of my favorite hobbies
7.1	He_goes_to_the library_every_Sunday
7.2	Every Sunday he goes to the library

To examine the usefulness of the rules that we extracted with the Stanford parser, we used the rules in different ways. In Table 3, we use C300, C450, C600, and C985, as the column headings, to indicate experimental results that we observed when we used 300, 450, 600, and 985 most frequent rules with the chart parser.

Except the very last row, the "I" column shows whether the sentence with a particular ID was accepted by the chart parser. We use "1" and "0" to indicate acceptance and rejection, respectively. The IDs are exactly the IDs shown in Table 2, except the last row. The "N" column shows the number of sentences (out of the *k*! permutations) that were accepted by the chart parser. The "SP" column shows the ranks of the sentences when we submitted the *k*! permutations to the Stanford parser.

Table 3. Using the chart parsers to filter sentences

ID	SP	C300		C450		C600		C985	
		I	N	I	N	I	N	I	N
1.1	25	1	32	1	78	1	96	1	240
1.2	12	1		1		1		1	
2.1	4	1	120	1	120	1	120	1	120
2.2	9	0		0		0		0	
2.3	10	1		1		1		1	
2.4	6	1		1		1		1	
6.1	1	0	0	0	0	0	0	0	0
6.2	19	0		0		0		0	
7.1	3	1	192	1	192	1	240	1	240
7.2	9	0		0		0		0	
10		0	0	12	4050	12	4050		

The very last row shows how the 48 sentences in Table 1 were accepted by the chart parsers. When we used only 300 rules, no sentences in Table 1 were accepted because of missing rules. When we used more than 300 rules, the parsers accepted only 12 sentences from Table 1, and accepted 4050 permutations (from 40320 permutations) as grammatical sentences.

The statistics in the "SP" column confirm that it is not easy to find the best α candidate sentences to capture the correct sentences. However, the chart parsers that used different numbers of rules did not always find the correct sentences either. The incorrect sentences could be accepted, and the correct sentences could be rejected. Increasing the number of rules allowed the parsers to accept more sentences, but may not help the acceptance of correct sentences.

To make the categorical parsers really useful, we may need to improve the grammar rules. This could be achieved by finding a better source of the rules – either by domain experts or by machine learning techniques.

6 A Heuristic: POS-based Pegging

If we do assume that only learners of the beginning levels will be interested in the practices of sentence reconstruction, we have a good reason to assume that the correct sentences are probably not very long and not very complicated. With these assumptions, we can do part-of-speech (POS) tagging of the correct sentence first, and peg words that have the same POS tags. Details about effectiveness and problems of using this heuristic will be discussed in an extended paper.

7 Summary

We have gone through an investigation of the techniques that are useful for the creation of test items for reconstructing scrambled sentences. It is not difficult to determine how to peg tokens in a set of candidate permutations to force unique solutions. It is not as easy to help the teachers to foresee all of the possible answers to the scrambled sentence tests. Generating the set of grammatical sentences efficiently has been a challenging task. We investigated the applications of the Stanford parser and a primitive categorical parser, and proposed a useful heuristic. Although the results are not conclusive, the current experimental results show encouraging improvements against the state of the art.

Acknowledgments

We were supported in part by the grant NSC-97-2221-E-004-007 from the National Science Council of Taiwan.

References

[1] Becker, T., Joshi, A. K., & Rambow, O. (1991). Long-distance scrambling and tree adjoining grammars, *Proc. of the 5th Conf. on European Chapter of the ACL*, 21–26.

[2] Burstein, J. & Leacock, C. Editors. (2005). *Proc. of the 2nd Workshop on Building Educational Applications Using Natural Language Processing*, The 43rd Annual Meeting of the ACL.

[3] Heift, T. & Schulze., M. (2007). *Errors and Intelligence in Computer-Assisted Language Learning*. NY, USA: Routledge.

[4] Joshi, A. K., Becker, T., & Rambow, O. C. (2002). Complexity of scrambling: A new twist to the competence-performance distinction. In Abeille, A. and Rambow, O. C., editors, *Tree-Adjoining Grammars*. Stanford: CSLI.

[5] Liu, C.-L. Lin, J.-H., & Huang, C.-B. (2008). A platform for authoring test items for elementary Chinese with techniques of natural language processing, presented in the 2008 CAERDA Int'l Conf.

[6] Manning, C. D. & Schultz, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press.

[7] Tetreault, J., Burstein, J., De Felice, R. Editors. (2008). *Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, The 46th Annual Meeting of the ACL

國科會補助專題研究計畫項下出席國際學術會議心得報告

日期：100年4月2日

計畫編號	NSC-97-2221-E-004-007-MY2		
計畫名稱	機率式建模技術與自然語言的標記、認知和教學		
出國人員姓名	劉昭麟	服務機構及職稱	國立政治大學資訊科學系教授
會議時間	2009年11月30日至2009年12月4日	會議地點	香港教育學院
會議名稱	（中文） （英文）The Seventeenth International Conference on Computers in Education		
發表論文題目	（中文） （英文）Computer assisted creation of items for scrambled sentence tests		

一、參加會議經過

ICCE 是一個有相當歷史的會議，早年的 ICCE 是國內的研討會，後來逐漸變成國際學術研討會，之後又歷經了幾年稿源比較短缺的時期，近幾年在中央大學數位老師的共同努力，透過國際合作又逐漸獲得許多人的重視。

透過中央大學陳德懷教授的引介，計畫主持人得以有機會邀集多位國際學者參與今年度 ICCE 的一個 interactive event: The Interactive Event on Applications of Virtual Agents, Student Modeling, and Knowledge Engineering in Education。一共有四位國際學者和五位國內學者參與報告：Mizoguchi 教授、Mitrovic 教授、Biswas 教授、Hirashima 教授、鄭淑真教授(南台科大)、林繼任教授(佛光大學)、陳志洪教授(新竹教大)、林豪鏘教授(台南大學)、黃永廣教授(雲科大)。這一特殊的會中會於十二月一日舉行。

除了召集、主辦與協調上述會中會之外，計畫主持人也另外發表一篇論文。這一篇論文主要是討論句子重組試題的電腦輔助出題技術。句子重組試題可以測驗語文學習者的文法知識，是很常見的測驗題型，要編輯好的句子重組試題，實際上不是一件容易的事情，本篇論文探討一些技術上不容易克服的問題與可能解決方案。

二、與會心得

ICCE 的主題是資訊科技在教學活動上的應用，比起計畫主持人近幾年都參加以計算語言學為主體的研討會來說，感覺上的親和性要好很多。講求實務的研討會雖然不一定必然有高深的技術議題，但是都是探索與真實應用問題息息相關的面向。這是參與 ICCE 這一類研討會最好的地方。透過這樣的研討會，才不至於流失於為技術而技術的錯誤。

ICCE 的論文有多種層面，有探討學生的學習模型者，有探討不同學科的教育軟體者，參與者有教育學家，有資訊軟硬體學家；展現了跨領域研究的特性。比起計畫主持人於 2008 年在美國所參加的 AERA 和 NCME 兩個年會性質有所不同。AERA 大多是教學學者來參與，可以聽到許多的教學經驗，NCME 則是測驗學家的會議。

截至目前為止，ICCE 或許只能算是還在復興階段，但是透過與會者的認真，個人相信這一個研討會將逐漸建立聲譽，成為電腦輔助教學領域中的一個重要會議。

在這一個會議之中遇到兩個來自政治大學教育學院結伴而來的女研究生，相對於我自己在(後來)2010 年無法找到學生到北京報告 COLING 論文，實在是覺得慚愧。

三、考察參觀活動(無是項活動者略)

無

四、建議

有一些領域其實不要在意論文的技術深度，而是要問那一些技術的應用意義。這樣才能夠掌握真實的應用。

五、攜回資料名稱及內容

六、其他

國科會補助專題研究計畫項下出席國際學術會議心得報告

日期：100 年 4 月 2 日

計畫編號	NSC-97-2221-E-004-007-MY2		
計畫名稱	機率式建模技術與自然語言的標記、認知和教學		
出國人員姓名	劉昭麟	服務機構及職稱	國立政治大學資訊科學系教授
會議時間	2010 年 08 月 23 日至 2010 年 08 月 27 日	會議地點	中國北京市
會議名稱	（中文） （英文）The Twenty Third International Conference on Computational Linguistics		
發表論文題目	（中文） （英文）Visually and phonologically similar characters in incorrect simplified Chinese words		

一、參加會議經過

因為計畫主持人於會議期間正在美國短期訪問，經費不足以專程回到亞洲與會，同時無法覓得適當研究生代為出席，因此沒有實際參與會議。論文的發表委由朝陽大學吳世弘教授代理。

二、與會心得

跟過去一樣，語言教學類別的研究問題，除非能夠涉及計算語言學的技術研究，如果只是應用型的研究議題，並不容易被收入會議長篇論文之列。當然，這並不意味著以計算語言學的技術來從事電腦輔助學習的價值受到貶抑。這只是意味著後者比較沒有直接的學術意義，但是仍有不小的實用意義。

三、考察參觀活動(無是項活動者略)

四、建議

五、攜回資料名稱及內容

六、其他

國科會補助專題研究計畫項下赴大陸地區出差報告

日期：100 年 4 月 2 日

計畫編號	NSC-97-2221-E-004-007-MY2		
計畫名稱	機率式建模技術與自然語言的標記、認知和教學		
出國人員 姓名	劉昭麟	服務機構 及職稱	國立政治大學資訊科學系教授
出國時間	2011 年 01 月 25 日至 2011 年 01 月 27 日	出國地點	中國北京市語言大學

一、 大陸研究過程

這一次到中國北京的語言大學交流，是一些巧合與善意的組合。計畫主持人在投稿 TALIP 論文的過程中，論文評審指出我們所進行的文獻回顧有重要的遺漏，漏掉了北京語言大學和上海交通大學的一些相關研究。由於計畫主持人在 2010 年下半年在美國密西根大學短期訪問，如果要透過網路去購買相關的文獻，必須先在中國擁有相關的帳號，因此不得其門而入。基於限時之內必須回應 TALIP 論文評審的修改要求，因此冒昧直接聯絡語言大學的資深院長宋柔教授索取論文，不意宋教授不只善意回應，並且於極短時間之內轉寄論文。雖然計畫主持人的 TALIP 論文最後獲得接受發表，不見得與得以閱讀和引用宋教授的相關著作絕對相關，但是這樣的協助與善意，加上研究工作的直接相關醞釀了計畫主持人於返國之後，再前往北京與宋教授面談的契機。

計畫主持人於一月五日晚上返回國內，結束半年的短期訪問，在收作整理之後，即申請前往北京語言大學訪問，於獲得國科會核可之後即前往北京。由於研究計畫的限制，同時農曆春節將屆，因此只暫留於北京語言大學一天。

計畫主持人於廿五日下午抵達北京，於廿六日訪問北京語言大學，於廿七日中午即搭機返國。

二、 研究成果

北京語言大學是北京市幾所大學中很特殊的大學，接受去中國學習中文的學生是主要的任務，去語言大學學習的學生不見得只是學習中文，還有可能是去北京語大學習特殊專業的，例如學習資訊科技的研究生。目前在北京語大學習的外國學生來自超過一百個國家，學生總人數接近一萬人。

北京語大的資訊學院規模不大，主要的任務是從事與語文學習相關的資訊科技，總共不及十位專業是資訊科技的教授，儘管如此，國內也少有類似規模專研語文教學相關科技的團隊。團隊之中，有專研語音科技、有專研機器翻譯科技、有專研書寫輔助科技的教授、甚至有專研中文斷詞科技的教授。不小的團隊把中文學習科技切分成數個非常小的個別領域。

計畫主持人此行主要是分享所發表的 TALIP 論文，除聽取相關的評論之外，並且與宋柔教授先前所做關於簡體漢字的類似研究相互交流。在交換意見之後，發現還有不少可以繼續深入探討的研究議題。

三、 建議

在中文學習成為極大市場過程中，台灣的產學界是否考慮以合作代替競爭與中國的產學界建立藍海關係？

四、其他