Enhancing Traditional Input Methods Through Part-of-Speech Tagging

Michael Brasser
Department of Computer Science
Calvin College
Grand Rapids, MI, 49546 USA
mbrass48@calvin.edu

Abstract

Inputting Chinese text into the computer is problematic because of the large number of characters. Those systems that have been developed either fail to take advantage of existing natural language processing techniques, or tend to relegate the user's role to that of post-editing. This paper proposes an input system that takes advantage of a part-of-speech tagger for contextual constraints, and involves the user during the conversion process.

Introduction

Inputting Chinese text into the computer is problematic because of the large number of characters. Historically, there have been three main approaches to solving this problem: code-based input, radical-based input, and pinyin-based input, each with its individual strengths and weaknesses. Consider the following sentence:

她	跑	得	很	快	。
She	runs		very	fast	.

(1)

To use a code-based system, the user must have previously memorized (or must look up) the specific code for each character. This method allows for extremely fast entry, but training can take as long as a year because of the daunting task of memorizing codes for large numbers of Chinese characters (Thompson, 1991). To use a radical-based system, the user need only remember the keystrokes for each radical in the Chinese language (there are around 250 of them). However, they must also be able to visualize how each radical is placed within the desired character. Pinyin-based input is perhaps the easiest to learn, as it is based on a phonetic spelling system for Chinese. The above example would be input as

ta pao de hen kuai. (2)

Although this phonetic input method is well suited for input on a standard keyboard of twenty-six letters, it has traditionally been one of the slowest, for reasons mentioned below. A good solution should be found to increase pinyin input speed. Yin (1991) describes three stages in the development of pinyin to character systems:

- syllable-based conversion
- whole words and phrase-based conversion
- sentence and paragraph-based conversion

The first stage (syllable-based conversion) consists of entering and converting Chinese characters one at a time. This requires prompting the user to choose the correct character after each pinyin 'word' has been entered, and is slow because of the large number of homonyms in the Chinese language (e.g., the "ta" in example (2) could represent any one of at least fourteen characters). The second stage (whole words and phrase-based conversion) allows the user to enter entire words or phrases in pinyin before converting to characters, and thus increases input speed and accuracy. However, it still requires prompting the user after each word or phrase. The third stage (sentence and paragraph based conversion) consists of converting whole paragraphs, or even pages, of pinyin at once. This is a natural continuation of the first two stages, and because conversion to characters is done automatically (without prompting the user), it allows for

increased input speed over stage one and stage two systems.

Most modern pinyin-to-Chinese character input systems are stage two systems. These systems have been greatly improved since their inception through the incorporation of word frequency and character bi-gram information, but still do not (and will not) provide an adequate solution to the input problem because they require user intervention after every word has been entered.

Research into developing stage three systems has been underway for at least ten years, with varying results. Lua (1992) describes two prototype systems: one which requires tonal information, and the other which does not. Accuracy rates for conversion were reported at 93.9% and 86.5% respectively. Several commercial products have also been introduced, including BOPOMOFO (2000) and KEY CJK (2000). In informal testing, both products exhibited accuracy rates of between eighty and ninety percent. For short compositions (100 characters or less), a ninety percent accuracy rate might be acceptable. But for large papers (2000+ characters), this would mean at least 200 (2000 * .10) errors made by the system. The problem is compounded by the fact that all of these systems relegate the users role to post-editor, requiring them to go back through the converted text to locate and correct the errors made by the system.

This paper describes RUPERT, a stage three system that attempts to improve pinyin input speed through the use of a part-of-speech (POS) tagger, and user involvement in the conversion process (when necessary). The POS tagger limits the set of possible characters for a pinyin string in context, and is appropriate in part because most current tagging algorithms yield a 96-97% accuracy rate (Jurafsky & Martin, 2000). While this does not automatically translate to a 96-97% total accuracy rate for the system, it provides a high accuracy base. In addition to tagging, RUPERT also involves the user during the conversion from pinyin to characters, much like a stage two system. While requiring too much user intervention could become wearisome, asking the user to choose the correct character when the system is unsure gives a significantly higher accuracy rate. Thus once the initial conversion is complete, there are very few errors for the user to locate and correct.

1 The Tagger

RUPERT uses QTAG (Mason & Tufis, 1998), a stochastic tagger based on the PARTS tagging algorithm (Church, 1988). I modified the tagger slightly to implement a more pure Hidden Markov Model algorithm before integrating it with RUPERT, as this gave better results.

QTAG was originally developed for English, but can be retrained for other languages by providing a pre-tagged training corpus. The corpus I chose for this task was a collection of twelve pre-tagged texts taken from elementary school readers in Singapore (Yu, Zhou, et al., 1996). The corpus is approximately 50,000 tokens in size, and contains over 2000 unique Chinese words. This was the largest freely available tagged corpus that I could locate, and proved useful for this project because it introduced the basic patterns and words of the Chinese language.

The corpus was converted from characters to pinyin using a pinyin annotator developed by Peterson (2000). As several Chinese characters have multiple pronunciations (for example, 大 can be pronounced “da” or “dai”), I hand-checked the text to insure correct conversion.

The tagset used to tag the corpus is described by Zhu et al. (1995). I modified the tagset slightly in order to give RUPERT more grammatical information when needed. In the original tagset, all three grammatical *de*’s were classified as “grammatical particle” (u). In keeping with their different usage, my modification gives each a unique category (的 (u), 地 (u2), and 得 (u3)). In addition, I manually decreased the probabilities of rare tags (prefix, suffix, shortened-name, etc). This was done because the relatively small size of the corpus, as well as its introductory nature, lent itself to over-representation of rare tags.

2 Pinyin Input

RUPERT’s pinyin input is governed by the following two principles:

1. The pinyin must be entered without tonal information. Although providing the

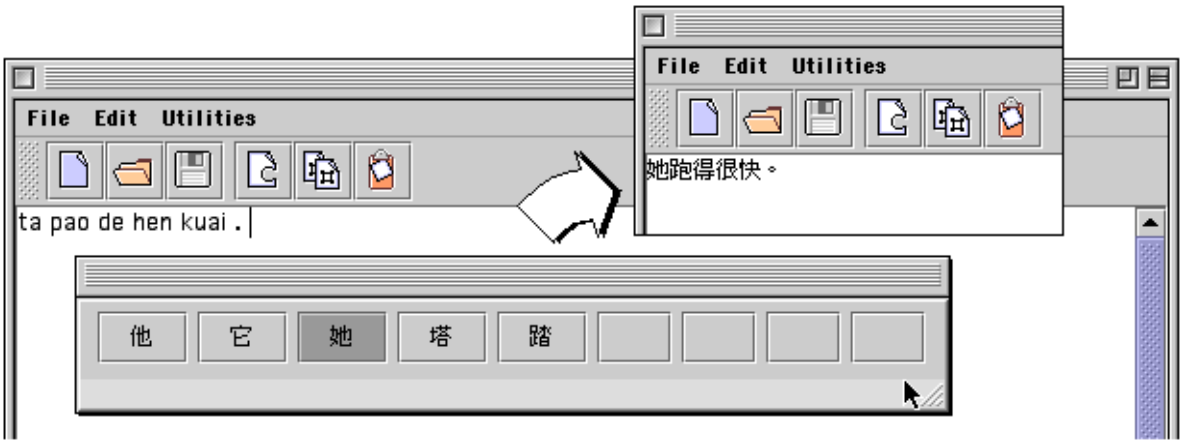

Figure 1 : RUPERT's User Interface

information allows for a higher accuracy rate (cf. Lua, 1992), it slows the user's typing speed by requiring them to type tonal indicators after each pinyin syllable.

2. The pinyin should be word segmented rather than syllable segmented. Although an automatic segmentation routine should eventually be added to the system, current pinyin input should conform to the word segmentation practiced in the training corpus.

3 Prompt Heuristics

When the user is ready to convert from pinyin to characters, the pinyin string is first analyzed by QTAG. Based on the information provided by the tagger, RUPERT must then decide whether to automatically display the most likely character for each pinyin word, or prompt the user to choose the correct character. There are several heuristics currently implemented by the system, based on P_{total}, an estimate of how accurate an assigned tag is, and N, the number of characters in a tag group with the same pronunciation. The formula for P_{total} is as follows:

$$P_{total} = \begin{cases} P_{first} / P_{any} & \text{if } P_{any} \neq 0 \\ 1 & \text{if } P_{any} = 0 \end{cases}$$

where P_{first} is the probability of the first tag occurring in the given context, and P_{any} is the probability of any of the word's tags occurring in the given context.

The guidelines, then, are as follows:

- For $P_{total} \geq .70$
  - If N = 1, the character is automatically inserted.
  - If N > 1, the user is prompted to choose the correct character.
- For $P_{total} < .70$, the user is prompted to choose the correct character.

Setting the threshold for P_{total} requires balancing the desires to catch the highest number of errors, while still keeping required user intervention to a minimum. If the threshold were set at 0, no mistakes would be made, but the user would be required to approve nearly every character, giving little advantage over a stage two system. If the threshold were set at 1, the user would never be asked to intervene, but the system would make more mistakes. The threshold was rather arbitrarily set to .70, which seemed to produce an optimal balance between the two desires.

4 Walkthrough

The steps taken by RUPERT to process example (1) are shown in figure 1. First, the user types in the pinyin sentence as shown in (2), and presses shift-enter to begin conversion. Tagging the sentence provides the following information[1]:

[1] Because of space constraints, only the first three tags (or less, if there are less than three tags) are shown for each word.

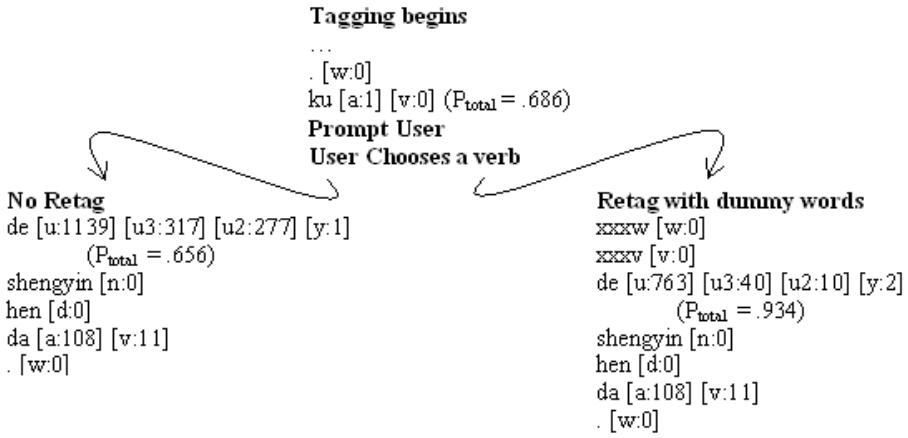

Figure 2 : Retagging the text

ta [r:1106] [n:0] [v:0] (P_{total} > .70, N > 1)
pao [v:0] (P_{total} > .70, N = 1)
de [u3:1751] [u:479] [u2:6] (P_{total} > .70, N = 1)
hen [d:0] (P_{total} > .70, N = 1)
kuai [a:43] [d:0] [n:0] (P_{total} > .70, N = 1)
. [w:0] (P_{total} > .70, N = 1)

The first word, "ta", has P_{total} > .70 and N > 1, so the user is prompted to choose the correct character. After the user indicates their choice, the rest of the characters appear automatically, without prompting the user, according to the rules outlined above. Note that the system converts as it goes along (the latter characters don't appear until the user has chosen the correct "ta").

5 Error Handling

In order to minimize the impact of errors, RUPERT examines which character the user chooses when prompted. If the character is not from the most probable tag group (i.e., in the tag ta[r:10] [v:4] [n:0], if the user chooses from any tag group other than 'r'), the system flushes it's tagging buffer and retags the text, taking into account the user's choice. This is done through dummy words, which were added manually to RUPERT's lexicon[2]. Two dummy words are inserted into the tagging buffer (representing the character just chosen by the user, as well as the character proceeding that), and tagging resumes. Because the second dummy word is guaranteed to have the correct tag (it was manually chosen by the user), the following text receives a more accurate tagging. Consider tagging the following example:

ku de shengyin hen da (3)
cry sound very big
"The crying is very loud."

As can be seen in figure 2, "ku" has been incorrectly tagged as an adjective. However, as P_{total} < .70, the user is prompted to choose the correct character. The left branch represents QTAG's continuing tag (after the user's choice) without the use of dummy words. In this case, the user is prompted to choose the correct "de", and then the rest of the tagging can proceed automatically. The right branch represents retagging with dummy words. Because the dummy word representing "ku" has the correct tag (v), the resulting tag of "de" is more accurate than that of the left branch, and it's P_{total} is pushed above the threshold. This allows tagging to proceed without the need of prompting the user to choose the correct "de".

6 Test Results

As RUPERT was trained on elementary school texts, several elementary-level texts were taken from a Chinese newsletter (Chen, 2000) to test the system, each ranging in size from 112 to 259 characters. For each of these texts, the accuracy of RUPERT's selections or suggestions was

[2] The tagset contains one dummy word for each of it's tags, and the dummy word is represented by *xxx* + the tag it represents. Thus, the dummy word for noun is *xxxn*, while the dummy word for verb is *xxxv*.

	Text One 194 words 242 characters	**Text Two** 191 words 259 characters	**Text Three** 88 words 112 characters	**Total** 473 words 613 characters
Automatic Insertion:				
auto-insertion rate	69%	74%	72%	72%
auto-insertion accuracy	96%	99%	100%	98
User Intervention:				
prompt rate	31%	26%	28%	28%
1st choice accuracy	78%	76%	76%	78%
Whole Text:				
Baseline accuracy	93.8%	95.8%	93.8%	94.6%
POS only accuracy	92.6%	95.4%	94.6%	94.1%
Final Accuracy	97.9%	99.6%	100%	99.0%

Figure 3 : RUPERT Accuracy Rates

recorded, and can be seen in figure 3. RUPERT's automatic insertion rate is shown in the first line of the figure ("auto-insertion rate"), and the accuracy of these insertions is shown in the second line ("auto-insertion accuracy"). The frequency with which the user was prompted is shown in the third line ("prompt rate"), and the accuracy of the first choice presented to the user is shown in the fourth line ("1st choice accuracy"). A baseline accuracy for each article, computed by inserting the most common character for each pinyin word, is shown on the fifth line ("Baseline Accuracy"). The accuracy of the system without any user intervention (i.e., if the P_{total} threshold were set to 1.0) is shown in the sixth line ("POS only accuracy"). The final accuracy of the system, taking into account both POS tagging and user intervention, is shown on the seventh line ("Final accuracy"). Any errors still present in the text at this point can be located and corrected by the user via a pop-up menu.

As seen in figure 3, RUPERT's POS only accuracy is not significantly different (and is in fact slightly lower) than the computed baseline accuracy. However, I suspect this is in large part a function of the limited, elementary-level domain used for training and testing RUPERT. Retraining and retesting RUPERT based on a larger, more-complex corpus should allow for a similarly high POS only accuracy, while the baseline accuracy would quickly drop for more complex examples. If so, RUPERT's POS only results would compare favorably to the 86.5% reported by Lua (1992) for a toneless system, and the averages of 80-90% obtained by BOPOMOFO and KEY CJK. In addition, RUPERT's reliance on a statistical POS tagger provides a mechanism for intelligent user intervention in a way the baseline system or a rule-based system, such as Lua's, do not. This allows for an even higher final accuracy rate of 99.0%.

7 Error Analysis

RUPERT's errors all occurred when tagging common, one-syllable words – i.e. "you", "de", "ba", and "na". This is not surprising, as most of these words have at least two commonly occurring tags, and as many as six possible tags. Furthermore, because HMM tagging operates on a buffer of only three words, there are cases where important contextual clues to a word's correct tag will exist outside of the buffer. Consider the following example:

ta pa de yidian ye bu kuai (4)
he crawl a little also not fast
"He crawls very slowly."

RUPERT incorrectly tagged the "de" as the wrong type of grammatical particle. The important clue needed ("kuai") was outside of the tagging buffer. Potential solutions for increasing accuracy are mentioned below.

Conclusion

This paper has described RUPERT, a pinyin input system based on POS tagging and user involvement during conversion. RUPERT uses POS tagging to narrow the space of possible characters for any pinyin spelling, and uses the probabilities produced by the POS tagger to determine when to prompt the user to make the final character selection. Taken together, these features allow the system a final accuracy rate of 99.0%, leaving very little to be post-edited by the user. However, several areas remain for further improvement:

- RUPERT's lexicon and probability base is too limited to be of real world use, and should be expanded.
- The concept of "word" in Chinese has traditionally been hard to define, so requiring the user to manually segment pinyin into words is problematic. Alternatively, an automatic word-segmenter could be added to the system. In addition, there has been much research into standardization of pinyin and it's segmentation (cf. State Language Commission, 1988), and this work could be used to improve RUPERT.
- A mechanism for processing unknown words should be added to RUPERT. During the testing, several words occurring in the texts did not exist in the lexicon, and needed to be replaced with linguistically equivalent words. For example, one text made frequent use of the word "信用卡" (credit card). This word was not in the lexicon, so it was replaced by the word "卡片" (card).
- The addition of other natural language processing components to the system (such as pronoun inferers) could reduce the prompt rate, without drastic effect on the final accuracy rate.
- Learning mechanisms to catch commonly committed errors would improve the overall accuracy of the system, while at the same time easing the frustration of users.

Acknowledgements

My thanks go to Keith Vander Linden for his help in revising the paper, Oliver Mason for allowing me to use and modify QTAG, and the reviewers for their helpful comments.

References

BOPOMOFO (2000) Canotec Communication Network. URL: http://www.canotec.co.jp/.

Chen, Kang Chu (editor) (2000) *Communications of the Chinese Association of Western Michigan.*

Church, K.W. (1998) A stochastic parts program and noun phrase parser for unrestricted text. In *Second Conference on Applied Natural Language Processing*, pp. 136-143. ACL.

Jurafsky, Daniel; Martin, James H. (2000) *Speech and Language Processing.* Prentice Hall.

KEY CJK (2000) Asia Communications Québec Inc. URL: http://www.cjkware.com/.

Lua, K.T.; Gan, K.W. (1992) A Touch-Typing Pinyin Input System. *Computer Processing of Chinese and Oriental Languages*, pp. 85-94.

Peterson, Erik (2000) On-line Chinese Tools. URL: http://www.mandarintools.com/.

State Language Commission, P.R.C. (1988) Basic Rules for Hanyu Pinyin Orthography.

Thompson, P.M. (1991) Chinese Text Input and Corpus Linguistics. *Characters and Computers*, IOS Press, pp. 122-130.

Tufis, Dan; Mason, Oliver (1998) Tagging Romanian Texts: a Case Study for QTAG, a Language Independent Probabilistic Tagger. Proceedings of *the First International Conference on Language Resources & Evaluation (LREC)*, Granada (Spain), 28-30 May 1998, pp.589-596.

Yin Binyong (1991) Pinyin-to-Chinese Character Computer Conversion Systems and the Realization of Digraphia in China. *Characters and Computers*, IOS Press, pp. 26-36.

Yu, Shiwen; Zhou, Qiang; Zhang, Wei; Zhang, Yunyun; Zhan, Weidong; Chang, Baobao; Sui, Zhifang (1996) Word Segmented and POS Tagged 12 Volumes of Singapore Chinese Primary School Text. *Communications of Colips*, p. 41.

Zhu, Xuefeng; Yu, Shiwen; Wang, Hui (1995) The Development of Contemporary Chinese Grammatical Knowledge Base and its Applications. *Communications of Colips.*