Intelligent Pinyin IME

Peng Wu

Content

- Pinyin IM Background
- OSS Pinyin IM Survey
- NLP-based Pinyin IM vs non-NLP
- sunpinyin vs novel-pinyin
- libpinyin - joint efforts
- libpinyin - rational
- libpinyin roadmap
- libpinyin goals

1. Pinyin IM Background

Pinyin IM Background

1. On Windows:
a. most Pinyin IME use sophisticated techniques (such as NLP...) to improve the correction rate of pinyin to characters conversion, without user manually choose every Chinese word.
b. but ABC Pinyin IM (likes ibus-table) is still there, some users refused to use new input approach.
2. On Linux:
a. We are improving, but there are still a long way to go.

2. OSS Pinyin IM Survey

OSS Pinyin IM Survey

1. Maximum Forward Match:
a. Fcitx
b. ibus-pinyin
2. Uni-gram(or word frequencies based.):
a. scim-pinyin
3. N-gram:
a. sunpinyin
b. novel-pinyin

Maximum Forward Match

Match steps:
1. match the first longest word and forward the cursor,
2. repeat above steps until all pinyin has been converted.

uni-gram

1. Statistical-based.
2. Match steps:
1. Try to find the sentence with the most frequent words.

Concrete Example

Example: zhong'guo'ren
P(中国人 |zhong'guo'ren)
= P(中国人)
= P(中国)*P(人)
= 0.001 * 0.001
= 1e-6

Concrete Example(Cont.)

P(种果人 |zhong'guo'ren)
= P(种果人)
= P(种果)*P(人)
= 0.0001 * 0.001
= 1e-7
< 1e-6 = P(中国人 |zhong'guo'ren)
So we will choose 中国人 as a result.

n-gram

1. Use more sophisticated Statistical Math Model.
2. Match steps:
Find the most possible sentence by using Statistical Language Model.
(n-gram will be explained later.)

3. NLP-based Pinyin IM vs non-NLP

A hypothesis pinyin IM

1. Record all Chinese pinyin/sentence pairs in a 2TB database.

Pros:

1. Nearly 100% correction rates.

Cons:

1. No such huge disk storage to store all pinyin/ sentence pairs.
2. No such powerful CPU can do a pinyin-sentence conversion in 1 second.

NLP-based Pinyin IM vs non-NLP

1. non-NLP based:
Logic is simple, but have problems on performance and correction rate.
2. NLP based:
Use mathematic models, and computing intensive, yields more correction rates; or yields similar correction rates with less computation time and less disk space.

4. sunpinyin vs novel-pinyin

sunpinyin overview

* Built on a back-off n-gram language model (3-gram)
* Supports multiple pinyin schemes (Quanpin & Double Pinyin)
* Supports fuzzy-segmentation, fuzzy-syllables, and auto-correcting.
* Supports 2-gram history cache, or user's customized model
* Ported to various OS/platforms: iBus, XIM, MacOS
* dual-licensed with CDDL+LGPLv2.1
* Applying the acceptance of Debian, available on Ubuntu, Fedora and Mandriva ...

sunpinyin intro

1. n-gram based algorithms.
2. Mathematic Model describes:
Try to find the maximum possibility sentence which can pronounces the given sentence.
3. In recent 3 years, sun-pinyin has been re-factored to easier be understood.

sunpinyin Math Model

To calculate the probability of sentence $S=(W_1,W_2,W_3...W_n)$

$P(S) = P(W_1).P(W_2|W_1).P(W_3|W_1,W_2).P(W_4|W_1,W_2,W_3)...P(W_n|W_1,W_2,...W_{n-1})$

$$P(S) = P(W_1)\prod_{i}^{n} P(W_i|W_1, W_2, ...W_{i-1})$$

In reality, due to the data sparseness, it's impossible to calculate the probability in this way. A particle method is, to assume the $P(W_i|W_1,W_2,...W_{i-1})$ only depends on the previous N words, i.e., $W_{i-N+1},W_{i-N+2},...W_{i-1}$.

Particularly, we have unigram (N=0 , context-free grammar), bigram (N=1), trigram (N=2), and fourgram (N=3). The most commonly used is bigram, trigram.

Concrete Example

Example: zhong'guo'ren
P(中国人 |zhong'guo'ren)
= P(中国人)
= P(中国)*P(人 | 中国)
= 0.01 * 0.1
= 0.001

Concrete Example(Cont.)

P(种果人 |zhong'guo'ren)
= P(种果人)
= P(种果)*P(人 | 种果)
= 0.01 * 0.01
= 0.0001
< 0.001 = P(中国人 |zhong'guo'ren)
So we will choose 中国人 as a result.

novel-pinyin overview

* Based on an interpolation smoothing of Hidden Markov Model. (bi-gram)
* Complete support for ShuangPin schemes, fuzzy pinyin and incomplete pinyin, inherited from scim-pinyin.
* Improved user input self-learning.
* Appears on some distro, SUSE, Mandriva, aur etc.

novel-pinyin intro

1. HMM-based (Hidden Markov Model) algorithms.
2. Mathematic Model describes:
Try to find the maximum possibility sentence which really pronounces the given sentence.
3. Clear interface design from the beginning, although algorithms are also complicated as sunpinyin.

novel-pinyin Math Model

$$H = \underset{H}{Argmax}\, P(H|P)$$

H stands for Hanzi sequences, P stands for pinyin sequences.
P(H|P) stands for the possibility of the corresponding Hanzi when Pinyin is given.

Math Model (Cont.)

$$\hat{H} = \underset{H}{\operatorname{argmax}} \frac{\Pr(P|H)\Pr(H)}{\Pr(P)} = \underset{H}{\operatorname{argmax}} \Pr(P|H)\Pr(H)$$

$$= \underset{H}{\operatorname{argmax}}\, p(w_1)\prod_{i=2}^{n} p(w_i|w_{i-1})\Pr(P|H)$$

$$= \underset{H}{\operatorname{argmax}}\, p(w_1)\prod_{i=2}^{n} p(w_i|w_{i-1})\, p(p_1 p_2 \cdots p_n | w_1 w_2 \cdots w_n)$$

$$= \underset{H}{\operatorname{argmax}}\, p(w_1)\prod_{i=2}^{n} p(w_i|w_{i-1})\prod_{i=1}^{n} p(p_i|w_i)$$

Concrete Example

Example: zhong'guo'ren
P(中国人 |zhong'guo'ren)
= P(中国人) *P(zhong'guo'ren| 中国人)
= P(中国)*P(人 | 中国)*P(zhong'guo| 中国)*P(ren| 人)
= 0.01 * 0.1 * 0.7 * 0.5
= 3.5*10^-4

Concrete Example(Cont.)

P(种果人 |zhong'guo'ren)
= P(种果人)*P(zhong'guo'ren| 种果人)
= P(种果)*P(人 | 种果)*P(zhong'guo| 种果)*P(ren| 人)
= 0.01 * 0.01 * 0.8 * 0.5
= 4.0*10^-5
< 3.5*10^-4 = P(中国人 |zhong'guo'ren)
So we will choose 中国人 as a result.

5. libpinyin - joint efforts

libpinyin - joint efforts

1. sunpinyin and novel-pinyin has been competing for 2-3 years.
2. when one input method finished some features, users will request the other one to implement such features. Combines the core part can save efforts to re-invent the wheels.
3. in the past two projects both have very limited developer resources, merging the core part will greatly alleviate this problem.

6. libpinyin - rational

libpinyin - rational

1. Unique library to process pinyin relative tasks for Chinese, just likes libchewing, libanthy.
2. NLP-based approach which provides higher corrections with reasonable speed and space cost.
3. Save efforts to avoid re-developing similar functionality when need to deal with pinyin or other Chinese tasks, just use libpinyin directly.

7. libpinyin roadmap

libpinyin roadmap

1. Compare features between ibus-pinyin, novel-pinyin and sunpinyin, try to figure out the super set of features.
2. Discuss various possible implementations of each component.
3. Pick up best component implementations from ibus-pinyin, novel-pinyin and sunpinyin, different implementations can co-exist.
4. Source code is compiling time configurable, by specifying the needed flags when compiling, this library can precisely simulates the behaviors of ibus-pinyin, novel-pinyin or sunpinyin.
5. Merge source codes, and coding more...

8. libpinyin goals

libpinyin goals

1. Code must be easily maintainable.
2. Consistent interface between components, each component can evolves independently.
3. Project code can make progress, without re-writing the entire code base for brand new idea.
4. Ability to include new NLP models, and work seamless with related components.
5. Consistent coding styles, and the most important thing is to ease code reading.
6. For hard algorithms, better to write in a separate module, instead to keep two similar algorithms together(eg, back-off and interpolation), to ease code reading. Everything which makes code ease to understand is welcome. :)

Q&A

Thanks