Automatic Dictionary Building

**Background**

In the field of Chinese natural language processing, word segmentation is a key step, which has a strong impact on subsequent steps. Though there are many general purposed toolkits that can cope with political or social news texts, when speaking of specialized fields like finance, the inner-set dictionary is quite limited.

In this weekly report, I would like to introduce an approach to build a dictionary from massive news texts.

**Problem**

Generally speaking, the dictionaries used for Chinese word segmentation (CWS) consist of thousands of words, altogether with their part of speech notation (POS) and frequency information, amongst which POS is sometimes omissible for some algorithms. The key problem for building up a new dictionary from massive untagged text is that the computer has no prior knowledge about words.

**Method**

GU Sen (<http://www.matrix67.com/blog/archives/5044>) purposed a simple but effective approach to fetch words from untagged text (互联网时代的社会语言学：基于SNS的文本数据挖掘, 程序员, 2012(7)-(8)). In this section, I shall introduce his method.

**Word in Definition.** Linguists define ‘word’ as ‘the smallest element that may be uttered in isolation with semantic or pragmatic content’. This deducts two important properties of words: first, the ‘smallest’, which means a word is not separable in languages; and second, ‘isolation’, means that a word is free in language, not bounded with other elements in language.

Inspired by the definition, Gu used probability to indicate how coherent a word candidate is, and information entropy to indicate how abundant the co-occurrence of a word is.

**Coherence.** Speaking of coherence (inseparability), let’s have a look at a three-character word ‘电影院’. Suppose that we have already known that ‘电影’ is a word. If ‘电影院’ is not, the probability of its occurrence may be estimated by multiplying that of ‘电影’ and ‘院’; if it is, its frequency will be much higher than this probability. Formally, we have:

If a substring c\_1 c\_2 … c\_n is a word, then we have p{c\_1} p{c\_2} … p{c\_n} << p{c\_1 c\_2 … c\_n}. The ‘<<’ relationship may be judged by the ratio of the two sides, comparing with a given threshold \theta\_p. (Prop. 1)

**Isolation.** Here information entropy is introduced to describe the isolation property of a word. For instance, let’s consider a substring ‘葡萄皮’, and two sentences ‘吃葡萄不吐葡萄皮’, ‘不吃葡萄倒吐葡萄皮’. We have ‘吐’ as the only left-neighborhood character for ‘葡萄皮’, but we have sets of characters {‘吃’ (probability = 1/2), ‘吐’ (1/2)}, and {‘不’ (1/4), ‘皮’ (1/2), ‘倒’ (1/4)} as its left- and right-neighborhood characters. Formally Denote C\_L as a set of all left-neighborhood characters and C\_R as that of all right-neighborhoods for a word W, we have the information entropy H(C\_R) and H(C\_L) both larger than a threshold \theta\_h. (Prop. 2)

**Definition.** We define that, a substring, which satisfies the consequences in (Prop. 1) and (2), is pragmatically a word.

**Implementation**

In Gu’s work, he suggested to store all text data in memory, and use indices to indicate substrings. In our cases, where near 23 GB of text is involved, such suggestion is inoperable.

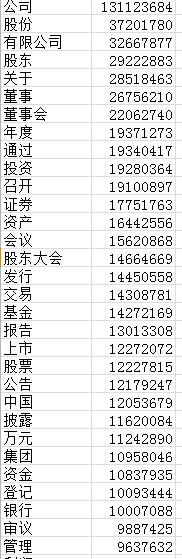
We have known that all probabilities involved in the computation is estimated from the samples’ word frequency. We may divide data into many files, which are big enough, but not too big, so that memory cost is endurable. We may assume that in each of these smaller files, those of very low frequency can be omitted.

In my implementation, I just scan each data file once, given the maximum word length M and file length N, the total processing time is approximately O(MN). The program (1) first reads the text file sentence by sentence, counts and stores all substrings shorter than M+2 in the sentence. (2) Then, it chooses those substrings shorter than M, occurring more than \theta\_t times, and meanwhile the frequency is \theta\_p times bigger than its coincidence probability, to build up a candidate word list. Each list item corresponds a C\_L and C\_R, got from the counting in step (1). (3) Finally, calculates the information entropy, and outputs those bigger than \theta\_h. After summing up all results from these files, we have got a word-frequency dictionary based on the given corpus.

**Experiments and Results**

The input data set is news for all 2412 stocks in Chinese market, fetched from Sina Finance, dating from July 2008 to July 2012.

The dictionary:



|  |  |
| --- | --- |
| Use default dictionary | Use dictionary generated from corpus |
| 20120813  快讯/银行/板块/午后/护/盘/交通银行/领/涨  新浪/财经/讯大蓝筹/在/地产/和/券商/拖累/下/集体/噤声/唯有/银行/板块/还/在/护/盘/午后/交通银行/领/涨/1/工商银行/平安/银行/农业银行/上涨/0.5/虽然/目前市场/对/银行/板块/中长期/的/前景/仍/难/报以/积极/的/预期/不过/随着/中/报/的/密集/披露/以及/大盘/本身/处于/企/稳/反弹/的/过程/中/银行/板块/在/近期/仍/存在/一定/的/机会 | 20120813  快讯/银行/板块午后/护盘/交通银行/领涨  新浪财经讯/大/蓝筹/在/地产/和/券商/拖累下/集体/噤/声/唯/有/银行板块/还/在/护盘/午后/交通银行/领涨/1/工商银行/平安银行/农业银行/上涨/0.5/虽然目前/市场对银行/板块/中长期/的/前景/仍难/报/以/积极/的/预期/不过/随着中报/的/密集/披露/以及/大盘/本身/处于/企稳反弹/的/过程中/银行板块/在近期/仍存在/一定的/机会 |

When speaking of proper noun and terms, our method performs better than the default dictionary. It recognized out all bank names and terms like ‘企稳反弹’, ‘蓝筹’, ‘护盘’ and ‘领涨’. However, the dictionary has a limited knowledge about common words or words in written language, like ‘唯有’ and ‘噤声’. There are also phrases like ‘新浪财经讯’, ‘虽然目前’, ‘市场对银行’ and ‘随着中报’; nonetheless, this finely reflects the fact that in Sina Finance’s news, such phrases are so fixed that we can regard them as words. By adding regular phrases into our dictionary, we may reduce noises resulted from stereotyped news writing.