Does Chinese Classical Poems from Tang Dynasty Conform to Rhyming Rules? A Quantitative Analysis

Lucius Hu

May 9, 2019

Intorduction

In Tang Dynasty of China, there emerged two types of rhyming poems, *LüShi* and *JueJu*, whose literal translations are *regulated verse* and *cut-off lines* respectively. The former one must have 8 sentences while the latter one shall have 4 sentences. Each sentence of either LüShi or JueJu shall have a fixed number of characters, either 5 or 7. Hence, those Tang poems are categorised into the followings according to the number of lines and the number of characters in each line:

- WuLü: 5-character 8-line regular verse
- QiLü: 7-character 8-line regular verse
- WuJue: 5-character quatrain
- QiJue: 7-character quatrain

Phoenetically, there is a complex set of rules regulating what characters could be allowed to be put into a certain position. Specifically, LüShi requires the last character of the 2nd, 4th, 6th, and 8th lines to be rhyming characters, and meanwhile JueJu requires last characters from the 2nd and 4th lines to rhyme with each other.

When writing a peom, its author not only needs to conform to the phoenetical rules, but also need to pay attention to the wording, and more importantly, exhibit its aethetical value. Thus it would take many years for a person to write a qualified poem, and only a very small number of poems ever written were good enough to be passed on through generations.

There's a collection of Tang Poems, *Three Hundred Tang Poems*, that contains 317 poems from Tang Dynasty. Among them, there are 227 poems that belong to the four aforementioned categories. It is interesting to know to what extent does these poems conform to the rhyming rules.

Specifically, there might be two types of reason that a poet would violate the rules. First, the author speaks a different variety of Chinese which experience some phoenetical change from the 'official language' during Tang Dynasty. Second, it's just very hard to find a rhyming characters and the author chose to sacrifice the phoenetics for better literature, and/or philosophical value.

Acquiring the Text Corpus of Three Hundred Tang Poems

The first data source is Wikisource, which has the full text body of Three Hundred Tang Poems. The table of contents of Three Hundred Tang Poems is available here. We will first extract the links to the four aforementioned categories of poems, and then download the html pages given the URLs. Then we will extract the text body from html source code using 'rvest' package.

```r
source('initialisation.R')
DT <- dlMenu()
str(DT[1])
```

```
## Classes 'data.table' and 'data.frame':   1 obs. of  4 variables:
##  $ type  : chr "Wulü"
##  $ author: chr "王勃"
##  $ title : chr "送杜少府之任蜀州"
##  $ url   : chr "https://zh.wikisource.org/wiki/%E9%80%81%E6%9D%9C%E5%B0%91%E5%BA%9C%E4%B9%8B%E4
##  - attr(*, ".internal.selfref")=<externalptr>
```

Most Wikisource pages we encountered has similar structures and most links to the poems are correct, thus there's not much difficulty for the text extraction. Specifically, each peom is in a '

'node, which has 4 child nodes with'

' tags (paragraph), which contains two sentences of LüShi or one sentence of JueJu.

Next, we just need to remove all empty lines and punctuations in the end of each sentences, which are one of the followings:

- ' ' (Halfwidth) Whitespace Character
- '　' Fullwidth White Space Character
- '，' Fullwidth Comma
- '。' Fullwidth Period
- '？' Fullwidth Question Mark
- '！' Fullwidth Exclamation Mark
- '；' Fullwidth Semicolon

In a small number of poems, there also contains footnotes, which takes one of the two following forms:

- '〈...〉' Footnotes between Fullwidth Opening Left-Angle Bracket and Fullwidth Right-Angle Bracket
- '[...]' Fottnotes between (Halfwidth) Left Bracket and (Halfwidth) Right Bracket

Below is a excerpt of codes we used to obtain the the text corpus.

```r
newtext <-
      read_html(f, encoding = 'UTF-8') %>%
      html_nodes('div.poem p') %>%
      html_text() %>%
      # must first remove new-line character so the other regex
      # would be correctly applied to multi-lines
      str_remove_all('\\n') %>%
      str_remove_all('〈.+?〉|\\[.+?\\]|　|，|。|？|！|；')
text <- c(text, newtext)
```

But instead of analysing the full text corpus, actually we need only the last characters of the even-numbered sentences. First, depends on the category to which a poem belong, the index of the characters we need to keep is different. The following lines show the way how we defined the four possible cases, with the 'switch' control flow:

```
idx <- function(type) {
    switch(type,
           Wulü  = seq(10, 40, 10),
           Qilü  = seq(14, 56, 14),
           Wujue = seq(10, 20, 10),
           Qijue = seq(14, 28, 14))
  }
```

Then, with the aide of 'purrr' package, we wrote a 'mapper' and apply each poems to it:

```
pmap(data[,.(type, text)],
       function(type, text) {
         nextChar <- character()
         for (i in idx(type)) {
           nextChar <- c(nextChar, str_sub(text, i, i))
         }
         nextChar <- paste0(nextChar, collapse = '')
       }
  )
```

Below shows how the data table look like, which contains the meta-data, i.e. type, author, title, as well as the urls, and both the full text and last characters we just extracted.

```
source('processPoems.R')
str(DT[1])
```

```
## Classes 'data.table' and 'data.frame':   1 obs. of  6 variables:
##  $ type    : chr "Wulü"
##  $ author  : chr "王勃"
##  $ title   : chr "送杜少府之任蜀州"
##  $ url     : chr "https://zh.wikisource.org/wiki/%E9%80%81%E6%9D%9C%E5%B0%91%E5%BA%9C%E4%B9%8B%
##  $ text    : chr "城🄵輔三秦風🄵望五津與君離🄵意同是宦🄵人海🄵存知己天涯若比🄵無爲在岐路兒女共
##  $ lastChar:List of 1
##   ..$ : chr "津人🄵巾"
##  - attr(*, ".internal.selfref")=<externalptr>
```

Acquiring the Rhymes of the Last Characters we extracted

At this point, we already get the set of characters, and we will need to get their rhymes. But first we will give a brief introduction to the second data source, from which we would obtained the rhymes.

https://ytenx.org is a website which allows the users to query phonetic attributes of a Chinese character, including the consonant, vowels, tones, definitions, and rhymes. Notice that rhymes and vowels are two related but distinct concept. If two characters have same vowels, they must rhyme with each other. But rhyming characters may have similar rather than identical vowels.

In Chinese Linguistic studies, the official language across the history are, *Old Chinese*, *Middle Chinese*, *Old Mandarin*, and *Mordern Mandarin*, in chronological order. The Old Mandarin and Modern Mandarin are both Mandarin language. But neither Old Chinese nor Middle Chinese is refering to *a* language, but rather *a family* of Chinese Languages.

The information available at https://ytenx.org are provided by volunteers, based on the authetic phoenetic dictionaries such as *上古音系*, *廣韻*, *洪武正韻牋*, *中原音韻*, and *分韻撮要*, which corresponds to Old Chinese, Middle Chinese, Sourthern Old Mandarin, Northern Old Mandarin, and Old Cantonese, respectively. During Tang Dynasty, the official language was Middle Chinese, and thus we need to get the data from 廣韻.

The following shows how the rhymes are extracted. First, similar to the extraction of text body, we initiate a HTML request and read the HTML source code of the response, and we use the html tag 'p yohn' and 'href' to find the link to the details page of a given character. On the detail page there's a HTML table, and there's a row named '韻攝', which contain the rhyme of the given character. And we will assign 'X' if the queried character is not found.

```r
rhm <-
      glue(
        'http://ytenx.org/zim?dzih={c}&kyonh=1&jtkb=1&jtdt=1&jtkd=1&jtgt=1'
      ) %>%
      read_html() %>%
      html_node('p.yonh a') %>%
      html_attr('href')
    if (!is.na(rhm)) {
      rhm <-
        glue('http://ytenx.org{rhm}') %>%
        read_html() %>%
        html_table() %>% .[[1]]
      rhm <-
        rhm[which(rhm$'X1' == '🄵攝'),2]
    } else {
      rhm <- 'X'
    }
```

Based on the rhymes we extracted, we now have a data frame where the first column is a character, and the second column is its rhyme. This data frame would serve us as a look-up table, and we will write a mapper to match each character to a rhyme according to the look-up table.

```
map(data,
      function(x) {
        nextRhyme <- character()
        for (i in unlist(str_split(x, ''))) {
          nextRhyme <- c(nextRhyme, dict[char == i]$rhymes)
        }
        nextRhyme <- paste0(nextRhyme, collapse = '')
      })
```

Here is an example of the matched characters and their rhymes for the first five lines of our data, where each line corresponds to a poem.

```
source('processRhymes.R')
head(DT[, .(lastChar, rhymes)], 5)
```

```
##    lastChar      rhymes
## 1: 津人🄵巾 臻 臻 臻 臻
## 2: 鵑泉田賢 山 山 山 山
## 3: 通中空翁 通 通 通 通
## 4: 陲知時期 止 止 止 止
## 5: 隅無殊夫 遇 遇 遇 遇
```

Exploratary Analysis

First, let's check the proportion of poems in each of the four categories that strictly conform to the rhyming rules. Based on the rhymes, we also appended a new columns of boolean variable, indicating whether the the rhyming characters are all having one identical rhyme.

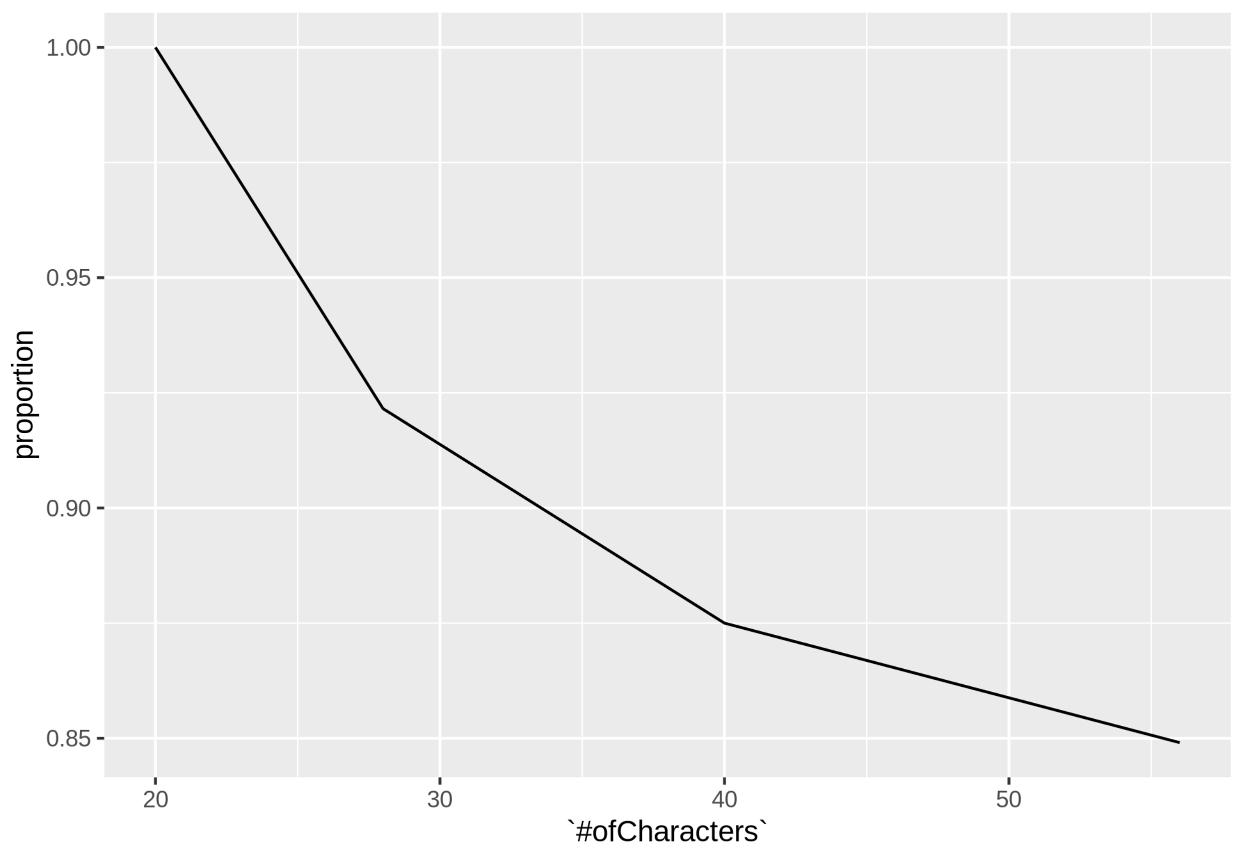
As we can see, it indeed seems easier to follow the rules when you only need to write four lines, as in Jueju, as oppose to LüShi, when you need to write eight lines. Secondly, for both LüShi and JueJu, it's easier to follow the rule when you the number of character is fewer.

```r
DT[, .(proportion = sum(.SD$correctRhyming)/nrow(.SD)), by = type]
```

```
##     type proportion
## 1:   Wulü  0.8750000
## 2:   Qilü  0.8490566
## 3: Wujue  1.0000000
## 4: Qijue  0.9215686
```

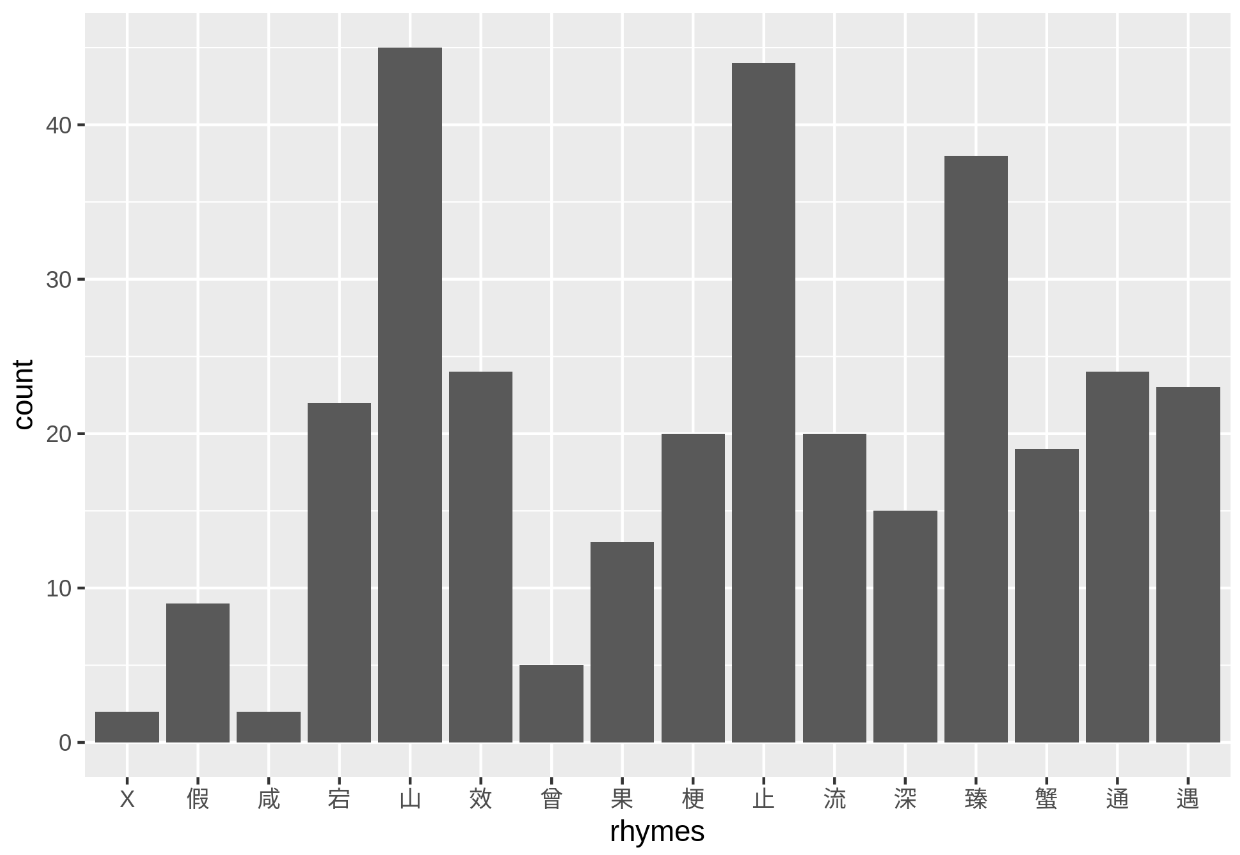
And if we plot such proportion against the number of character, than we will have the following visualisation. And obviously, there is a negative correlation between the total number of characters in a poem, and the tendency that it would strictly conform to the rhyming rules. This coincides with our gut sense that more characters imposes a greater difficulty on rhyming.

```r
p <- paste0(tempfile(),'.png')
ggsave(p,
       ggplot(
         DT[ ,
             .(proportion = sum(.SD$correctRhyming)/nrow(.SD)),
             by = type][ ,
                         '#ofCharacters' := c(40, 56, 20, 28)]) +
    geom_line(aes(y = proportion, x = `#ofCharacters`)))
grid.raster(readPNG(p,'.png'))
```



Last, we may also check the what rhymes are more often used.

```
p <- paste0(tempfile(),'.png')
ggsave(p, plot = ggplot(DTRhymes) + geom_histogram(aes(rhymes), stat = 'count'))
grid.raster(readPNG(p,'.png'))
```



count
40
30
20
10
0
X
假
咸
宕
山
效
曾
果
梗
止
流
深
臻
蟹
通
遇
rhymes