Stupid Poet Development Document

# Training Model

The model is formed by n-grams of a collection of Chinese poets. 2-grams are trained for in-sentence word continuation and cross-sentence word continuation. 3-grams are trained for only in-sentence word continuation.

蒹葭苍苍

白露为霜

Cross-sentence 2-gram

In-sentence 2-gram

# Generating Model

The sentences are generated by following steps:

1. The first word of the sentence is generate by cross-sentence 2-gram.
2. The second word of the sentence will firstly try being generated by in-sentence 2-gram. If failed, then use cross-sentence 2-gram.
3. The next word is generate by cross-sentence 2-gram or in-sentence 2-gram. It is more likely to choose in-sentence 2-gram.
4. The next word will firstly try being generated by in-sentence 2-gram. If failed, then use cross-sentence 2-gram.
5. Repeat step 4 and step 5 except the last word of the sentence.
6. The last word is generate by cross-sentence 2-gram, in-sentence 2-gram or in-sentence 3-gram.
7. If the sentence is the third one or seventh one, the last word will firstly be generated by in-sentence 2-gram for 4-word sentences, or in-sentence 3-gram for 5-or-7-word sentences. If failed, then use cross-sentence 2-gram.
8. Otherwise, the last word is generated by in-sentence 2-gram or cross-sentence 2-gram. It is more likely to choose cross-sentence 2-gram.

朝辞白帝彩云间

千里江陵一日还

Cross-sentence 2-gram

In-sentence 2-gram

In-sentence 3-gram

Must generate

More likely generate

Less likely generate

# Model Storage

The model is consists of N-grams. The N-grams consist of following fields:

* Content: The N-word content.
* Occurrance: How many times the N-gram occurs.
* Type: The type of N-gram. Valid values: “in-sentence”, “cross-sentence”.

The N-grams are stored as json. The following illustrates an example:

|  |
| --- |
| {  "content": "一日还",  "occurance": 1,  "type": "in-sentence"  } |