Language Modeling

Knersy-Ney Smoothing & KenLM

by Rong Zhou

Language Modeling review

-Why need it: e.g. “I am going to bed” and “I am going to bad”

-What it can do: sign probability for a sentence $S = w_1, w_2, \ldots, w_m.$

n-gram:

$$P(S) = \prod_{t=1}^{m} P(w_t | w_{1:t-1}) \quad (1)$$

$$P(w_t | w_{1:t-1}) = \frac{C(w_{1:t})}{C(w_{1:t-1})} \quad (2)$$

$$P(S) = \prod^{m} P(w_i | w_{i-n+1:i-1}) \quad (3)$$

$$P(w_t | w_{t-n+1:t-1}) = \frac{C(w_{t-n+1:t})}{C(w_{t-n+1:t-1})} \quad (4)$$

-Category: frequence based:

- n-gram model

neural network based:

- feedforward model (n-gram based)
- recurrent neural network model

Absolute discounting smoothing:

$$d \in [0, 1]$$

$$P_{abs}(w_i|w_{i-n+1:i-1}) = \frac{\max(C(w_{i-n+1:i}) - d, 0)}{C(w_{i-n+1:i-1})} + \lambda(w_{i-n+1:i-1}) P_{abs}(w_i|w_{i-n+2:i-1}) \quad (6)$$

$\lambda(w_{i-n+1:i-1})$ is the penalty for the lower order ngram $w_{i-n+2:i}$.

Knersy-Ney Smoothing (KN):

-backoff the bigram”reading _” to its unigram “_”

eg. I cannot see without my reading _.

-continuation smoothing:

$$P_{con}(w_i|w_{i-n+2:i-1}) = \frac{|\{w : C(w, w_{i-n+2:i}) > 0\}|}{|\{w_{i-n+1:i} : C(w_{i-n+1:i}) > 0\}|} \quad (7)$$

-continuation probability:

$$P(w_i|w_{i-n+1:i-1}) = \frac{\max(C(w_{i-n+1:i}) - d, 0)}{C(w_{i-n+1:i-1})} + \lambda(w_{i-n+1:i-1}) P_{con}(w_i|w_{i-n+2:i-1}) \quad (8)$$

In real world senario, $P_{con}(w_i|w_{i-n+2:i-1})$ may still be zero! So we can further improve Equation 8 and write it in a recursive fashion:

$$P_{KN}(w_i|w_{i-n+1:i-1}) = \frac{\max(C_{KN}(w_{i-n+1:i}) - d, 0)}{C_{KN}(w_{i-n+1:i-1})} + \lambda(w_{i-n+1:i-1})P_{KN}(w_i|w_{i-n+2:i-1}) \tag{9}$$

$$C_{KN}(\bullet) = \begin{cases} C(\bullet) & \text{for the highest order} \\ |\{w : C(w, \bullet) > 0\}| & \text{for the lower order} \end{cases} \tag{10}$$

Where

$$\lambda(w_{i-n+1:i-1}) = \frac{d}{C(w_{i-n+1:i-1})}|\{w : C(w_{i-n+1:i-1}, w) > 0\}| \tag{11}$$

KenLM

Hash tables and PROBING:

- $O\left(\frac{m}{m-1}\right)$ mption:

$$(96m + 64)c_1 + 128m \sum_{n=2}^{N-1} c_n + 96mc_N. \quad (12)$$

Stored Arrays and TRIE:

-time consumption:

$$O(\log\log|A|)$$

-memory consumption:

$$(32+32+64+64)c_1+ \sum_{n=2}^{N-1}(\lceil\log_2 c_1\rceil+q+r+\lceil\log_2 c_{n+1}\rceil)c_n+ (\lceil\log_2 c_1\rceil+q)c_N \quad (13)$$

Experimental result consist with the hypothesis

Package	Variant	Time (m) CPU	Time (m) Wall	RAM (GB) Res	RAM (GB) Virt
Ken	PROBING-L	72.3	72.4	7.83	7.92
	PROBING-P	73.6	74.7	7.83	7.92
	TRIE-L	80.4	80.6	4.95	5.24
	TRIE-P	80.1	80.1	4.95	5.24
	TRIE-L 8[a]	79.5	79.5	3.97	4.10
	TRIE-P 8[a]	79.9	79.9	3.97	4.10
SRI	Default	85.9	86.1	11.90	11.94
	Compact	155.5	155.7	9.98	10.02
IRST	Cache-Invert-L	106.4	106.5	5.36	5.84
	Cache-Invert-R	106.7	106.9	5.73	5.84
	Invert-L	117.2	117.3	5.27	5.67
	Invert-R	117.7	118.0	5.64	5.67
	Default-L	126.3	126.4	5.26	5.67
	Default-R	127.1	127.3	5.64	5.67
Rand	Backoff[a]	277.9	278.0	4.05	4.18
	Backoff[b]	247.6	247.8	4.06	4.18

Ref.Kenneth Heafield

Summary

The performance of KN is very close to the state of the art. That is due to its continuation probability calculation; The contribution of KenLm mainly lies in the two kinds of data structure: Hash Table & PROBING and Sorted Arrays & TRIE. All of them laid the fundamention for language modeling.