Bounded-memory Language Model Building

Ivan Pouzyrevsky, Mohammed Mediani, Kenneth Heafield

September 8, 2012

Pipeline

1. Count 5-grams
2. Adjust counts
3. Compute uninterpolated probabilities
4. Interpolate probabilities
5. Compute backoff sums and merge with probabilities

Pipeline

1. Count 5-grams
2. Adjust counts
3. Compute uninterpolated probabilities
4. Interpolate probabilities
5. Compute backoff sums and merge with probabilities

Sorting: Suffix Order

Suffix Order

5	4	3	2	1
A	A	A	A	A
C	A	A	A	A
A	Y	A	B	A
A	Y	Z	B	A
A	A	A	A	Y
C	A	B	A	Z

Sorting: Suffix Order

Suffix Order

5	4	3	2	1
A	A	A	A	A
C	A	A	A	A
A	Y	A	B	A
A	Y	Z	B	A
A	A	A	A	Y
C	A	B	A	Z

Context Order

4	3	2	1	5
A	A	A	A	A
A	A	A	A	Y
C	A	A	A	A
C	A	B	A	Z
A	Y	A	B	A
A	Y	Z	B	A

Sorting: Suffix Order

Suffix Order

5	4	3	2	1
A	A	A	A	A
C	A	A	A	A
A	Y	A	B	A
A	Y	Z	B	A
A	A	A	A	Y
C	A	B	A	Z

Context Order

4	3	2	1	5
A	A	A	A	A
A	A	A	A	Y
C	A	A	A	A
C	A	B	A	Z
A	Y	A	B	A
A	Y	Z	B	A

Sorts are performed with TPIE:
https://github.com/thomasmoelhave/tpie

Pipeline

1. Count 5-grams → suffix sort
2. Adjust counts → context sort
3. Compute uninterpolated probabilities → suffix sort
4. Interpolate probabilities → context sort
5. Compute backoff sums and merge with probabilities

Pipeline

1. Count 5-grams → suffix sort
2. Adjust counts → context sort
3. Compute uninterpolated probabilities → suffix sort
4. Interpolate probabilities → context sort
5. Compute backoff sums and merge with probabilities

Avoid two sorts by using memory $\propto$ vocabulary size

Adjusted Counts

$$c(w_1^n) = \begin{cases} \#(w_1^n) & \text{if } n = 5 \text{ or } w_1 = \text{<s>} \\ |\{v : vw_1^n \in \text{model}\}| & \text{otherwise} \end{cases}$$

Discounts

$$D(c) = \begin{cases} 0 & \text{if } c = 0 \\ D_1 & \text{if } c = 1 \\ D_2 & \text{if } c = 2 \\ D_{3+} & \text{if } c \geq 3 \end{cases}$$

Interpolating

$$p_{KN}(w_i \mid w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i}) - D(c(w_{i-n+1}^{i}))}{\sum_{w_i} c(w_{i-n+1}^{i})} + \gamma(w_{i-n+1}^{i-1})\, p_{KN}(w_i \mid w_{i-n+2}^{i-1})$$

where

$$\gamma(w_{i-n+1}^{i-1}) = \frac{D_1\, N_1(w_{i-n+1}^{i-1}\bullet) + D_2\, N_2(w_{i-n+1}^{i-1}\bullet) + D_{3+}\, N_{3+}(w_{i-n+1}^{i-1}\bullet)}{\sum_{w_i} c(w_{i-n+1}^{i})}$$

Backoffs

$$B(w_1^n) = \frac{1 - \sum_{v:w_1^n v \in \text{model}} p(v \mid w_{1}^n)}{1 - \sum_{v:w_1^n v \in \text{model}} p(v \mid w_{2}^n)}$$

Use MapReduce! [Brants+, 2007]

Limitation of MapReduce: one input stream, one output stream.
w_1^n needs to speak with $w_2^n \implies$ hard to shard.

See BigFatLM for a Hadoop implementation
http://github.com/jhclark/bigfatlm