深蓝学院
shenlanxueyuan.com

第七章：语言模型 作业讲评

主讲人 俞帆

纲要

- 第一部分：SRILM
- 第二部分：插值语言模型公式推导
- 第三部分：哥大E6870

SRILM

- Kaldi中包含了SRILM工具，无需额外下载

- 使用-help查看工具提供的各个可选参数

SRILM

- 数据准备

- 生成ARPA语言模型（参考实现）：

```
ngram-count -text 训练文本 -order 3 -limit-vocab -vocab 词典 -unk
-map-unk “<SPOKEN_NOISE>” -wbdiscount -interpolate -lm 语言模型文件
```

- 计算PPL（参考实现）：

```
Ngram -ppl 测试文本 -order 3 -lm 语言模型文件
```

纲要

- 第一部分：SRILM
- 第二部分：插值语言模型公式推导
- 第三部分：哥大E6870

插值语言模型公式推导

深蓝学院 shenlanxueyuan.com

插值平滑的递归形式：

递归至一元 $P(w_i)$ → ML / uniform

$$P_{interp}(w_i|w_{i-N+1}^{i-1}) = \lambda_{w_{i-N+1}^{i-1}} P_{ML}(w_i|w_{i-N+1}^{i-1}) + (1-\lambda_{w_{i-N+1}^{i-1}}) P_{interp}(w_i|w_{i-N+2}^{i-1})$$

N元　N元　N-1元

更新公式：held-out 集合上更新入，train

①E步：expected count

$$E(\lambda_{w_{i-N+1}^{i-1}}) = \sum_{w_i} c(w_{i-N+1}^{i}) \frac{\lambda_{w_{i-N+1}^{i-1}} * P_{ML}(w_i|w_{i-N+1}^{i-1})}{\lambda_{w_{i-N+1}^{i-1}} P_{ML}(w_i|w_{i-N+1}^{i-1}) + (1-\lambda_{w_{i-N+1}^{i-1}}) P_{interp}(w_i|w_{i-N+2}^{i-1})}$$

长为N的字串在dev出现的次数

②M步

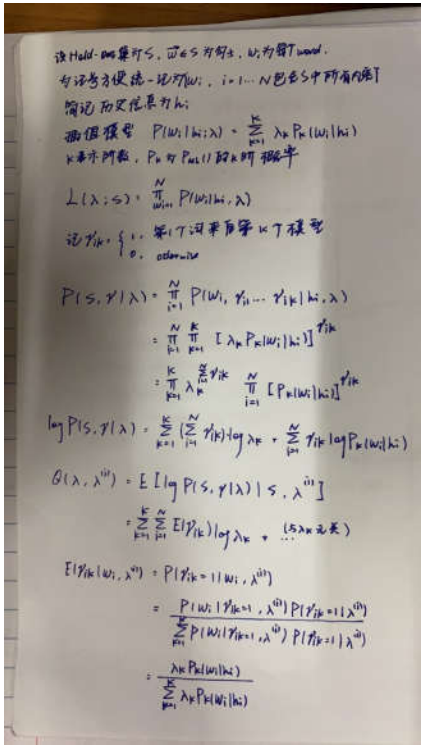
$$\tilde{\lambda}_{w_{i-N+1}^{i-1}} = \frac{E(\lambda_{w_{i-N+1}^{i-1}})}{\sum_{j=1}^{N} E(\lambda_{w_{i-N+j}^{i-1}})}$$

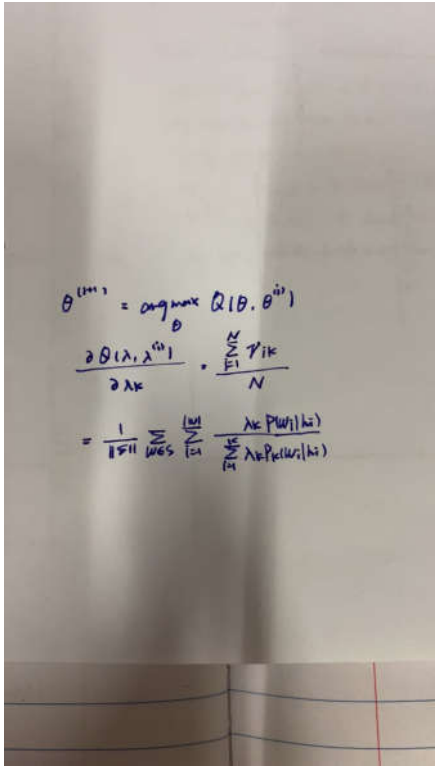
$\sum_{\lambda} = 1$

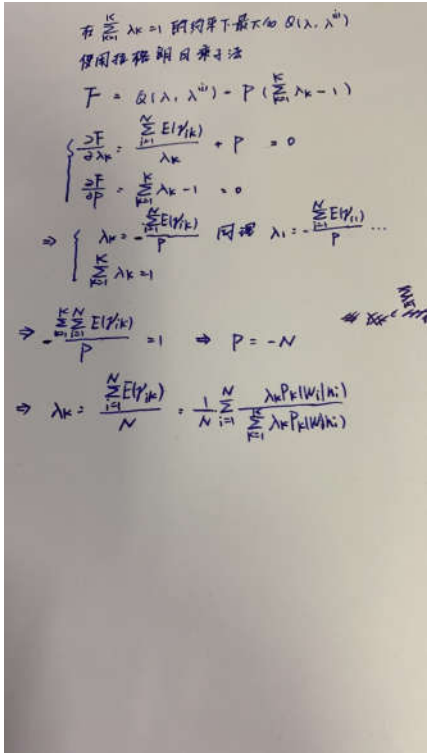
更新至 $|\lambda - \tilde{\lambda}| < \varepsilon$ 或 steps数 > n，stop。

http://www.cs.jhu.edu/~hajic/courses/cs465/cs46506/ppframe.htm

插值语言模型公式推导







纲要

- 第一部分：SRILM
- 第二部分：插值语言模型公式推导
- 第三部分：哥大E6870

NGramCounter Class Reference

Class for storing counts for a set of n-grams. More...

`#include <util.H>`

List of all members.

Public Member Functions

	NGramCounter () *Ctor; initializes object to be empty.*
void	**write** (ostream &outStrm, const **SymbolTable** &symTable=**SymbolTable**()) const *Writes all counts to stream `outStrm` in a text format.*
void	**clear** () *Clears object; deletes all n-grams in table.*
unsigned	**size** () const *Returns number of n-grams in table.*
bool	**empty** () const *Returns whether object is empty.*
unsigned	**incr_count** (const vector< int > &ngram) *Increments count of an n-gram; returns new count.*
void	**set_count** (const vector< int > &ngram, unsigned val) *Sets count of an n-gram to `val`.*
unsigned	**get_count** (const vector< int > &ngram) const *Returns count of an n-gram, or 0 if not present.*

哥大E6870

- 实现ngram计数

对lang_model.c 中的 count_sentence_ngrams 进行补全。

```
// static map<vector<int>, set<int> > histOnePlusMap;
for (int wordIdx = m_n - 1; wordIdx < wordCnt; ++wordIdx) {
  // process from n-gram to 1-gram
  for (int n = 1; n <= m_n; ++n) {
    // process m_predCounts
    vector<int> ngram(wordList.begin() + wordIdx - (m_n - n),
                      wordList.begin() + wordIdx + 1);
    int count = m_predCounts.incr_count(ngram);
    // process m_histCounts
    vector<int> histNgram(ngram.begin(), ngram.end() - 1);
    m_histCounts.incr_count(histNgram);
    if (count == 1) {
      m_histOnePlusCounts.incr_count(histNgram);
    }
  }
}
```

哥大E6870

- 实现Witten-Bell平滑算法

$$P_{\mathrm{WB}}(w_i|w_{i-1}) = \frac{c_h(w_{i-1})}{c_h(w_{i-1}) + N_{1+}(w_{i-1})} P_{\mathrm{MLE}}(w_i|w_{i-1}) + \frac{N_{1+}(w_{i-1})}{c_h(w_{i-1}) + N_{1+}(w_{i-1})} P_{\mathrm{backoff}}(w_i)$$

$$P_{\mathrm{backoff}}(w_i) = P_{\mathrm{WB}}(w_i) = \frac{c_h(\epsilon)}{c_h(\epsilon) + N_{1+}(\epsilon)} P_{\mathrm{MLE}}(w_i) + \frac{N_{1+}(\epsilon)}{c_h(\epsilon) + N_{1+}(\epsilon)} \frac{1}{|V|}$$

哥大E6870

深蓝学院
shenlanxueyuan.com

```
vector<int> histNgram(ngram.begin(), ngram.end() - 1);
int predCnt = m_predCounts.get_count(ngram);
int histCnt = m_histCounts.get_count(histNgram);
int histOnePlusCnt = m_histOnePlusCounts.get_count(histNgram);

double lambda = 0.0, PMle = 0.0, beta = 1.0, PBackoff;
if (histCnt > 0) {
  lambda = 1.0 * histCnt / (histCnt + histOnePlusCnt);
  PMle = 1.0 * predCnt / histCnt;
  beta = 1.0 * histOnePlusCnt / (histCnt + histOnePlusCnt);
}
if (ngram.size() == 1) {
  // recursive terminate
  PBackoff = 1.0 / vocSize;
} else {
  // recursive
  PBackoff =
      get_prob_witten_bell(vector<int>(ngram.begin() + 1, ngram.end()));
}
retProb = lambda * PMle + beta * PBackoff;
```

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening