Processamento Estatístico da Linguagem Natural

Aula 10
Professora Bianca
(Sala 302 – Bloco E)
bianca@ic.uff.br
http://www.ic.uff.br/~bianca/peln/

Aula de Hoje

- Cap. 4 – Jurafsky & Martin
  - Seções 4.5.1, 4.5.2, 4.6, 4.7 (menos 4.7.1), 4.8, 4.10

Técnicas de "smoothing" melhores

- Intuição usada por muitos algoritmos de smoothing como
  - Good-Turing
  - Kneser-Ney
  - Witten-Bell

  é usar a contagem de eventos que vimos apenas uma vez para estimar a contagem de eventos que nunca vimos.

Good-Turing

- Imagine que você está pescando
  - Existem 8 espécies: carpa, bagre, traíra, truta, salmão, enguia, tilápia, corvina
- Você pescou
  - 10 carpas, 3 bagres, 2 traíras, 1 truta, 1 salmão, 1 enguia
- Qual é a probabilidade da próxima espécie ser nova (i.e. tilápia ou corvina)
  - 3/18
- Qual é a probabilidade da nova espécie ser truta?
  - Menos de 1/18

Good-Turing

- Notação: N_x é a o número de itens com freqüência x (freqüência-da-freqüência)
  - Logo no exemplo N_{10}=1, N_1=3, etc
- Para estimar o número total de espécies
  - Usar o número de espécies (ou palavras) que vimos uma vez
  - $c_0^* = c_1$ $p_0 = N_1/N = 3/18$
- Todas as outras contagens são ajustadas para baixo de acordo com a seguinte fórmula:

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

c(enguia) = c*(1) = (1+1) 1/ 3 = 2/3

Good-Turing

	unseen (bass or catfish)	trout
c	0	1
MLE p	$p = \frac{0}{18} = 0$	$\frac{1}{18}$
c^*		$c^*(\text{trout}) = 2 \times \frac{N_2}{N_1} = 2 \times \frac{1}{3} = .67$
GT p^*_{GT}	$p^*_{\text{GT}}(\text{unseen}) = \frac{N_1}{N} = \frac{3}{18} = .17$	$p^*_{\text{GT}}(\text{trout}) = \frac{.67}{18} = \frac{1}{27} = .037$

Freqüências-de-freqüências e estimativas de Good-Turing

AP Newswire			Berkeley Restaurant—		
c (MLE)	N_c	c^* (GT)	c (MLE)	N_c	c^* (GT)
0	74,671,100,000	0.0000270	0	2,081,496	0.002553
1	2,018,046	0.446	1	5315	0.533960
2	449,721	1.26	2	1419	1.357294
3	188,933	2.24	3	642	2.373832
4	105,668	3.24	4	381	4.081365
5	68,379	4.22	5	311	3.781350
6	48,190	5.19	6	196	4.500000

Good-Turing - complicações

- Na prática, é comum supôr que as contagens grandes (c>k para algum k) são confiáveis:

$$c^* = c \text{ for } c > k$$

- Isso complica c*:

$$c^* = \frac{(c+1)\frac{N_{c+1}}{N_c} - c\frac{(k+1)N_{k+1}}{N_1}}{1 - \frac{(k+1)N_{k+1}}{N_1}}, \text{ for } 1 \le c \le k.$$

- Também: supor que contagens c=1 não são confiáveis, então tratamos N-gramas com contagem 1 como se fossem N-gramas com contagem zero.
- Também, precisamos que os N_k sejam não-nulos, então temos que interpolar as contagens N_k antes de computar c* a partir deles.

"*Backoff*" e Interpolação

- Se estamos estimando a probabilidade do trigrama p(z|xy) mas c(xyz) é zero.
  - Podemos usar informação do bigrama p(z|y).
  - Ou até do unigrama p(z).
- Como combinar unigramas, bigramas e trigramas?

"*Backoff*" vs. Interpolação

- "***Backoff***" : usar trigrama se a contagem for não-nula, senão usar bigrama se a contagem for não-nula, senão usar unigrama.
- **Interpolação**: média ponderada do trigrama, bigrama e unigrama.
  - Pesos são escolhidos usando um corpus de validação (hold-out).

Interpolação

- Interpolação simples

$$\begin{aligned}\hat{P}(w_n|w_{n-1}w_{n-2}) &= \lambda_1 P(w_n|w_{n-1}w_{n-2}) \\ &+\lambda_2 P(w_n|w_{n-1}) \\ &+\lambda_3 P(w_n)\end{aligned}$$

- Lambda's dependendo do contexto:

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1(w_{n-2}^{n-1})P(w_n|w_{n-2}w_{n-1}) \\ &+\lambda_2(w_{n-2}^{n-1})P(w_n|w_{n-1}) \\ &+\lambda_3(w_{n-2}^{n-1})P(w_n)\end{aligned}$$

Katz Backoff

$$P_{\text{katz}}(w_n|w_{n-N+1}^{n-1}) = \begin{cases} P^*(w_n|w_{n-N+1}^{n-1}), & \text{if } C(w_{n-N+1}^{n}) > 0 \\ \alpha(w_{n-N+1}^{n-1})P_{\text{katz}}(w_n|w_{n-N+2}^{n-1}), & \text{otherwise.} \end{cases}$$

$$P_{\text{katz}}(z|x,y) = \begin{cases} P^*(z|x,y), & \text{if } C(x,y,z) > 0 \\ \alpha(x,y)P_{\text{katz}}(z|y), & \text{else if } C(x,y) > 0 \\ P^*(z), & \text{otherwise.} \end{cases}$$

$$P_{\text{katz}}(z|y) = \begin{cases} P^*(z|y), & \text{if } C(y,z) > 0 \\ \alpha(y)P^*(z), & \text{otherwise.} \end{cases}$$

P* é a probabilidade "descontada" com GT

Intuição do backoff+desconto

- Não podemos usar estimativas MLE para a probabilidade no Backoff porque elas não somariam 1.
- Quanta probabilidade devemos atribuir aos trigramas com contagem 0?
  - Usar GT ou outro algoritmo de smoothing
  - As estimativas de nível mais baixo (N-1) servirão para dar pesos diferentes dependendo do contexto.
- O que fazemos quando houverem palavras com contagem 0 no corpus de teste?
  - **For do vocabulário** = palavras OOV = usar <UNK>

Questões Práticas

- Normalmente trabalhamos com o log das probabilidades
  - Serve para evitar underflow
  - Soma é mais rápida que multiplicação

$$p_1 \times p_2 \times p_3 \times p_4 = \exp(\log p_1 + \log p_2 + \log p_3 + \log p_4)$$

Formato ARPA

- N-gramas com Backoff normalmente são armazenados no formato ARPA:

unigram:	$\log p^*(w_i)$	w_i	$\log \alpha(w_i)$
bigram:	$\log p^*(w_i \mid w_{i-1})$	$w_{i-1}w_i$	$\log \alpha(w_{i-1}w_i)$
trigram:	$\log p^*(w_i \mid w_{i-2}, w_{i-1})$	$w_{i-2}w_{i-1}w_i$	


```
\data\
ngram 1=1447
ngram 2=9420
ngram 3=5201

\1-grams:
-0.8679678    </s>
-99           <s>                        -1.068532
-4.743076     chow-fun                   -0.1943932
-4.266155     fries                      -0.5432462
-3.175167     thursday                   -0.7510199
-1.776296     want                       -1.04292
...

\2-grams:
-0.6077676    <s>   i                    -0.6257131
-0.4861297    i     want                 0.0425899
-2.832415     to    drink                -0.06423882
-0.5469525    to    eat                  -0.008193135
-0.09403705   today </s>
...

\3-grams:
-2.579416     <s>   i         prefer
-1.148009     <s>   about     fifteen
-0.4120701    to    go        to
-0.3735807    me    a         list
-0.260361     at    jupiter   </s>
-0.260361     a     malaysian restaurant
...
\end\
```


Voltando à Perplexidade

- Qual é a dificuldade de reconhecer os dígitos ‘0,1,2,3,4,5,6,7,8,9,oh’: fácil, perplexidade 11
- Qual é a dificuldade de reconhecer (30.000) nomes de empregados da Microsoft. Difícil: perplexidade = 30.000
- Se o sistema tem que reconhecer
  - Operador (1 em 4 ligações)
  - Vendas (1 em 4 ligações)
  - Suporte técnico (1 em 4 ligações)
  - 30000 nomes (1 em 120000 ligações cada)
  - Perplexidade é 54
- Perplexidade é o fator de ramificação ponderado

Slide from Josh Goodman

Detalhes avançados

- Perplexidade é tecnicamente
  - “a exponenciação da entropia cruzada do modelo nos dados de teste”
  - Para entender isso, temos que aprender um pouco de teoria da informação

Entropia de Shannon

- A entropia de uma variável aleatória X é:

$$H(X) = -\sum_{x \in \chi} p(x) \log_2 p(x)$$

- Exemplo:
  - Mandar um telegrama dizendo em que cavalo apostar, de 8 cavalos
  - Podemos usar 3 bits
  - Então na média, se apostamos o dia inteiro, estaremos mandando:
    - 3 bits/aposta para especificar o cavalo
  - Podemos mandar menos bits?

Podemos usar menos bits!

- Suponha que a distribuição de cavalos em que estamos apostando seja:

Horse 1	$\frac{1}{2}$	Horse 5	$\frac{1}{64}$
Horse 2	$\frac{1}{4}$	Horse 6	$\frac{1}{64}$
Horse 3	$\frac{1}{8}$	Horse 7	$\frac{1}{64}$
Horse 4	$\frac{1}{16}$	Horse 8	$\frac{1}{64}$

- A entropia = melhor código seria:

$$\begin{aligned} H(X) &= -\sum_{i=1}^{i=8} p(i) \log p(i) \\ &= -\tfrac{1}{2}\log\tfrac{1}{2} - \tfrac{1}{4}\log\tfrac{1}{4} - \tfrac{1}{8}\log\tfrac{1}{8} - \tfrac{1}{16}\log\tfrac{1}{16} - 4(\tfrac{1}{64}\log\tfrac{1}{64}) \\ &= 2 \text{ bits} \end{aligned}$$

Entropia de sequências

- Entropia de uma sequência

$$H(w_1, w_2, \ldots, w_n) = -\sum_{W_1^n \in L} p(W_1^n) \log p(W_1^n)$$

- Taxa de entropia (entropia por palavra)

$$\frac{1}{n} H(W_1^n) = -\frac{1}{n} \sum_{W_1^n \in L} p(W_1^n) \log p(W_1^n)$$

Para medir a entropia de uma linguagem

- Precisamos pensar em sequências de tamanho infinito.
- Considere uma linguagem com um processo estocástico L que gera palavras:

$$\begin{aligned} H(L) &= \lim_{n \to \infty} \frac{1}{n} H(w_1, w_2, \ldots, w_n) \\ &= \lim_{n \to \infty} \frac{1}{n} \sum_{W \in L} p(w_1, \ldots, w_n) \log p(w_1, \ldots, w_n) \end{aligned}$$

- Teorema de Shannon-McMillan-Breiman:

$$H(L) = \lim_{n \to \infty} -\frac{1}{n} \log p(w_1 w_2 \ldots w_n)$$

Entropia Cruzada

- Dada uma distribuição $\boldsymbol{p}$, e o nosso modelo dessa distribuição $\boldsymbol{m}$,
- A entropia cruzada de $\boldsymbol{m}$ em $\boldsymbol{p}$ é:

$$H(p, m) = \lim_{n \to \infty} \frac{1}{n} \sum_{W \in L} p(w_1, \ldots, w_n) \log m(w_1, \ldots, w_n)$$

- Versão do teorema de Shannon-McM-Breiman:

$$H(p, m) = \lim_{n \to \infty} -\frac{1}{n} \log m(w_1 w_2 \ldots w_n)$$

Entropia cruzada

- Para qualquer modelo $\boldsymbol{m}$

$$H(p) \leq H(p, m)$$

  i.e. a entropia cruzada é um limite superior para a entropia.
- E quanto melhor $\boldsymbol{m}$, mais próxima será a entropia cruzada da entropia H(p)
- Então podemos usar a entropia cruzada para comparar modelos.
- O melhor modelo tem a menor entropia cruzada.

Finalmente: perplexidade

- Se temos um modelo M = P(W):

$$H(W) = -\frac{1}{N}\log P(w_1 w_2 \ldots w_N)$$

$$\begin{aligned}\text{Perplexity}(W) &= 2^{H(W)} \\ &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}} \\ &= \sqrt[N]{\prod_{i=1}^{N}\frac{1}{P(w_i|w_1 \ldots w_{i-1})}}\end{aligned}$$

Comparando perplexidades

- Minimizar perplexidade = maximizar probabilidade do conjunto de teste de acordo com o modelo.
- Exemplo:
  - Perplexidade de N-gramas
  - V = 20K
  - Treinamento = 38M palavras
  - Teste = 1.5 M palavras

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Google N-Gram Release

All Our N-gram are Belong to You

By Peter Norvig - 8/03/2006 11:26:00 AM

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects, such as statistical machine translation, speech recognition, spelling correction, entity detection, information extraction, and others. While such models have usually been estimated from training

http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

Google N-Gram Release

to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

Google N-Gram Release

- serve as the incoming 92
- serve as the incubator 99
- serve as the independent 794
- serve as the index 223
- serve as the indication 72
- serve as the indicator 120
- serve as the indicators 45
- serve as the indispensable 111
- serve as the indispensible 40
- serve as the individual 234