Processamento de Linguagem Natural

Modelagem de Linguagens

Prof.: Hansenclever Bassani (Hans) hfb@cin.ufpe.br
Site da disciplina: www.cin.ufpe.br/~hfb/pln/

Baseado nos slides do curso de Stanford no Coursera
por Daniel Jurafsky e Christopher Manning.

Tradução: Ygor Sousa
Revisão: Hansenclever Bassani

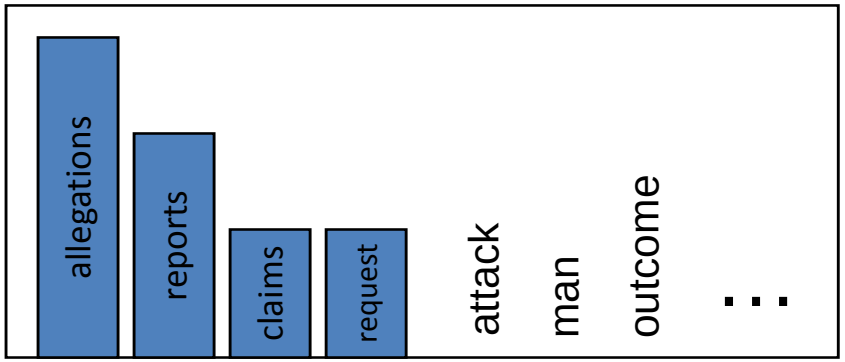
Modelagem de Linguagens

Suavização: Add-one ou (Laplace) Smoothing

A Intuição de Suavização (por Dan Klein)

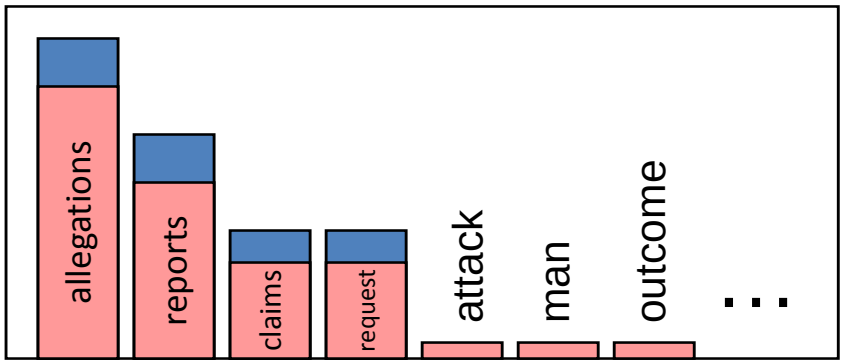
- Quando temos estatísticas esparsas:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request
7 total



- Roubar massa de probabilidade para generalizar melhor

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total



Estimação Add-one

- Também chamada de suavização de Laplace (smoothing)
- “Finge” que nós vimos cada palavra uma vez a mais do que realmente vimos
- Apenas adicione um a todas as contagens!
- Estimativa MLE:

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

- Estimativa Add-1:

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)+1}{c(w_{i-1})+V}$$

Estimativas de Máxima Verossimilhança

- A estimativa de máxima verossimilhança (MLE)
  - de alguns parâmetros de um modelo M de um conjunto de treinamento T
  - maximiza a verossimilhança do conjunto de treinamento T dado o modelo M
- Suponha que a palavra “bagel” aparece 400 vezes em um montante de um milhão de palavras
- Qual é a probabilidade de que uma palavra aleatória de algum outro texto será “bagel”?
- A estimativa MLE é 400/1,000,000 = .0004
- Esta pode ser uma má estimativa para algum outro corpo de texto
  - Mas esta é a **estimativa** que torna **mais provável** que “bagel" ocorra 400 vezes em um texto de um milhão de palavras.

Textos do Restaurante de Berkeley: Contagens Bigram com Suavização de Laplace

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Bigrams com Suavização Laplace

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Contagens Reconstituídas

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n)+1] \times C(w_{n-1})}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Comparação Original x Nova

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Estimação Add-1 é um instrumento cego

- Assim add-1 não é usado para N-grams:
  - Nós veremos métodos melhores
- Mas add-1 é usado para suavizar outros modelos PLN
  - Para Classificação de Texto
  - Em domínios onde o número de zeros não é tão grande.

Modelagem de Linguagens

Suavização: Add-one ou (Laplace) Smoothing

Modelagem de Linguagens

Interpolação, Recuo e LMs de Escala Web

UNIVERSIDADE FEDERAL DE PERNAMBUCO

Recuo e Interpolação

- As vezes ajuda a usar **menos** contexto
  - Condição em menos contexto para contextos em que não se aprendeu muito sobre
- **Recuo:**
  - Usar trigram se você tiver boas evidências,
  - Se não, usar bigram ou unigram
- **Interpolação:**
  - Misturar unigram, bigram, trigram
- Interpolação funciona melhor

Interpolação Linear

- Interpolação Simples

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

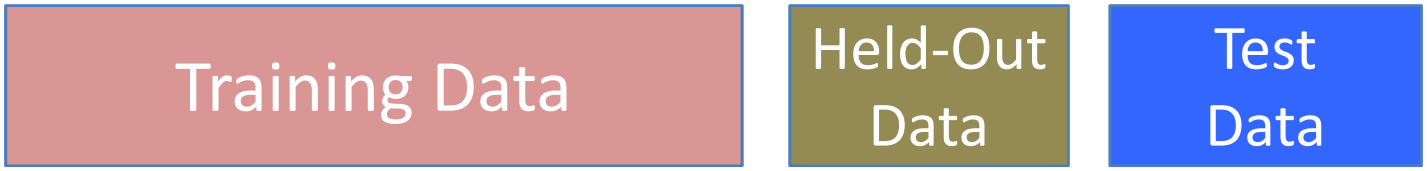
$$\sum_i \lambda_i = 1$$

- Lambdas condicionais no contexto:

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

Como escolher os lambdas?

- Utilizar um conjunto de **resistência (held-out)**



- Escolher λs para maximizar a probabilidade dos dados held-out:
  - Encontrar as probabilidades N-gram (nos dados de treinamento)
  - Posteriormente, procurar por λs que dão a maior probabilidade para o conjunto de held-out:

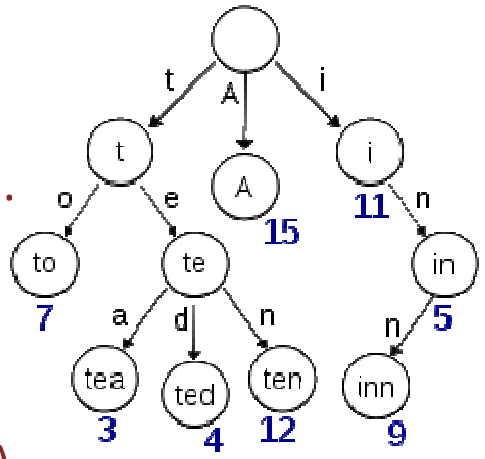
$$\log P(w_1 \ldots w_n \mid M(\lambda_1 \ldots \lambda_k)) = \sum_i \log P_{M(\lambda_1 \ldots \lambda_k)}(w_i \mid w_{i-1})$$

Palavras desconhecidas: Tarefas de vocabulário Abertas x Fechadas

- Se sabemos todas as palavras com antecedência
  - Vocabulário V é fixado
  - Tarefa de Vocabulário Fechada
- Sempre que não sabemos
  - **Out Of Vocabulary (Fora do vocabulário)** = palavras OOV
  - Tarefa de Vocabulário Aberta
- Ou então: criar um token de palavra desconhecida <UNK>
  - Obtenção de probabilidades de <UNK> (Treinamento)
    - Criar um vocabulário léxico L de tamanho V
    - Na fase de normalização do texto, qualquer palavra que não esta em L muda para<UNK>
    - Agora nós obtemos (através de treinamento) suas probabilidades como uma palavra normal
  - No momento de descodificação
    - Se entrada de texto: Usar probabilidades de UNK para qualquer palavra que não esteja em treinamento

N-grams de Escala Web (Alta Escala)

- Como lidar com, e.g., Coleção Google de N-gram
- Poda
  - Apenas armazenar N-grams com contagem > threshold.
    - Remover singletons de n-grams de maior ordem
  - Poda baseada em Entropia
- Eficiência
  - Estrutura de dados eficientes como árvores radix (tries)
  - Filtros Bloom: aproximar modelos de linguagem
  - Armazenar palavras como índices, não strings
    - Usar codificação Huffman para encaixar um grande número de palavras em dois bytes
  - Quantificar probabilidades (4-8 bits ao invés de 8-byte float)



Suavização para N-grams de Escala Web

- "Recuo estúpido (Stupid backoff)" (Brants *et al*. 2007)
- Não ligue, apenas use frequências relativas

$$S(w_i \mid w_{i-k+1}^{i-1}) = \begin{cases} \dfrac{\text{count}(w_{i-k+1}^{i})}{\text{count}(w_{i-k+1}^{i-1})} & \text{if} \quad \text{count}(w_{i-k+1}^{i}) > 0 \\ 0.4\,S(w_i \mid w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

$$S(w_i) = \frac{\text{count}(w_i)}{N}$$

Resumo: Suavização N-gram

- Suavização Add-1:
  - Bom para categorização de texto, não para modelagem de linguagens
- O método mais comumente utilizado:
  - Kneser-Ney Estendido e Interpolado
- Para N-grams muito grandes como os Web:
  - Stupid backoff

Modelagem de Linguagem Avançada

- Modelos Discriminativos:
  - Escolher pesos n-gram para melhorar uma tarefa, não para ajustar o conjunto de treinamento
- Modelos Baseados em *Parsing*
- Modelos de *Caching*
  - Palavras usadas recentemente são mais prováveis de aparecerem

$$P_{CACHE}(w \mid history) = \lambda P(w_i \mid w_{i-2} w_{i-1}) + (1-\lambda)\frac{c(w \in history)}{|history|}$$

  - Tem desempenho pobre para reconhecimento de discurso (Por quê?)

Modelagem de Linguagens

Interpolação, Recuo e LMs de Escala Web

UNIVERSIDADE FEDERAL DE PERNAMBUCO

Modelagem de Linguagens

Avançado:

Suavização Good-Turing

Lembrete: Suavização Add-1 (Laplace)

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)+1}{c(w_{i-1})+V}$$

Formulações mais gerais: Add-k

$$P_{Add-k}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + k}{c(w_{i-1}) + kV}$$

$$P_{Add-k}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + m(\frac{1}{V})}{c(w_{i-1}) + m}$$

Suavização Prévia Unigram

$$P_{Add-k}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + m(\frac{1}{V})}{c(w_{i-1}) + m}$$

$$P_{\text{UnigramPrior}}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + mP(w_i)}{c(w_{i-1}) + m}$$

Algoritmos Avançados de Suavização

- Linha de pensamento usada por muitos algoritmos de suavização
  - Good-Turing
  - Kneser-Ney
  - Witten-Bell
- Use a contagem de coisas que vimos **uma vez**
  - para ajudar a estimar a contagem de coisas que **nunca vimos**

Notação: N_c = Contagem de frequência de c

- N_c = contagem de coisas que vimos c vezes
- Sam I am I am Sam I do not eat

```
I    3
sam  2
am   2
do   1
not  1
eat  1
```

$N_1 = 3$

$N_2 = 2$

$N_3 = 1$

Intuição da Suavização Good-Turing

- Você está pescando (um cenário de Josh Goodman) e pega:
  - 10 carpas, 3 percas, 2 sardinhas, 1 truta, 1 salmão, 1 enguia = 18 peixes
- Qual a probabilidade da próxima espécie ser truta?
  - 1/18
- Quão provável é a próxima ser uma nova espécie (i.e. bagre ou baiacu)
  - Vamos usar nossa estimativa de coisas-que-vi-uma-vez para estimar novas coisas.
  - 3/18 (Sendo, N_1=3)
- Considerando isso, quão provável é a nova espécie ser truta?
  - Precisa ser menos de 1/18
  - Como estimar?

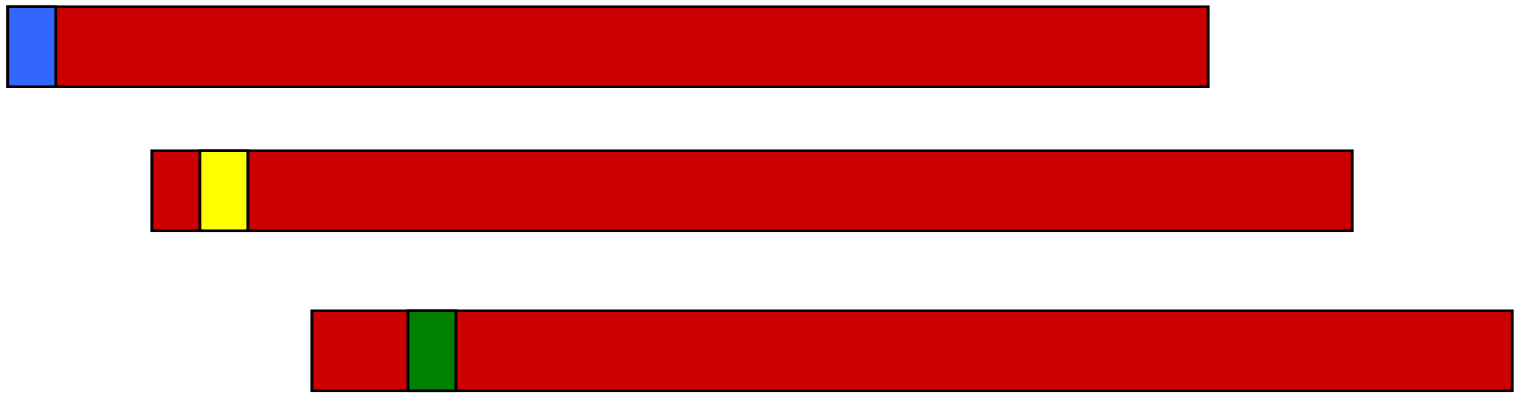
Cálculos Good Turing

$$P^*_{GT}(\text{coisas com frequência zero}) = \frac{N_1}{N} \qquad c^* = \frac{(c+1)N_{c+1}}{N_c}$$

- Não visto (baiacu ou bagre)
  - c = 0:
  - MLE p = 0/18 = 0
  - P^*_{GT} (Não visto) = N_1/N = 3/18
- Visto uma vez (truta)
  - c = 1
  - MLE p = 1/18
  - C*(truta) = 2 * N2/N1
    = 2 * 1/3
    = 2/3
  - P^*_{GT}(truta) = (2/3)/18 = 1/27

Intuição de Good-Turing - Ney et al.

H. Ney, U. Essen, and R. Kneser, 1995. On the estimation of 'small' probabilities by leaving-one-out. IEEE Trans. PAMI. 17:12,1202-1212

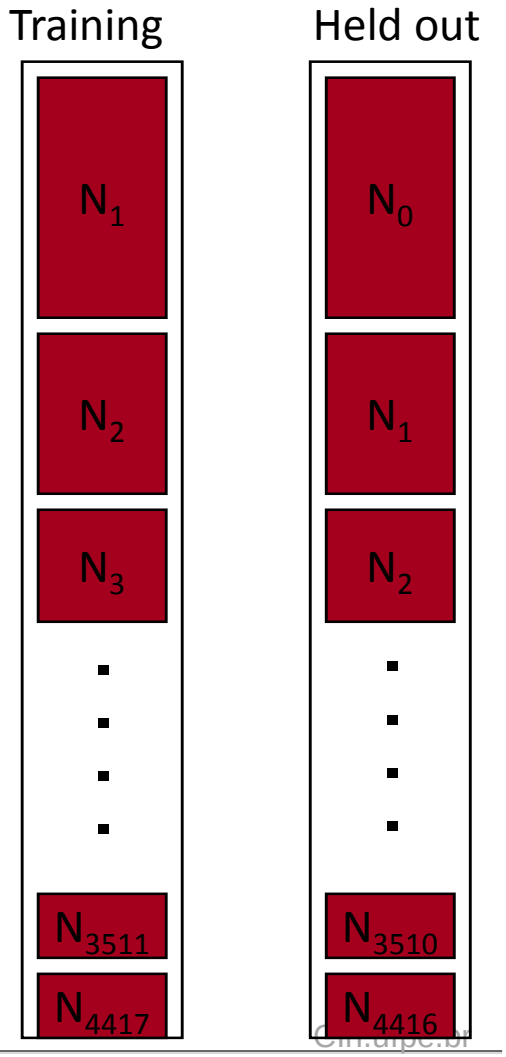


Palavras Held-out:

Intuição de Good-Turing - Ney et al. (slide de Dan Klein)

- Intuição da validação deixe-um-fora
  - Tire cada uma das *c* palavras de treinamento por turno
  - *c* conjuntos de treinamento de tamanho *c*–1, held-out de tamanho 1
  - Que fração de palavras held-out não são vistas em treinamento?
    - N_1/c
  - Que fração de palavras held-out são vistas *k* vezes em treinamento?
    - $(k+1)N_{k+1}/c$
  - Assim, no futuro, esperamos $(k+1)N_{k+1}/c$ das palavras serem aquelas com contagem de treinamento *k*
  - Existem N_k palavras com contagem de treinamento *k*
  - Cada um deve ocorrer com probabilidade:
    - $(k+1)N_{k+1}/c/N_k$
  - ...ou contagem esperada:

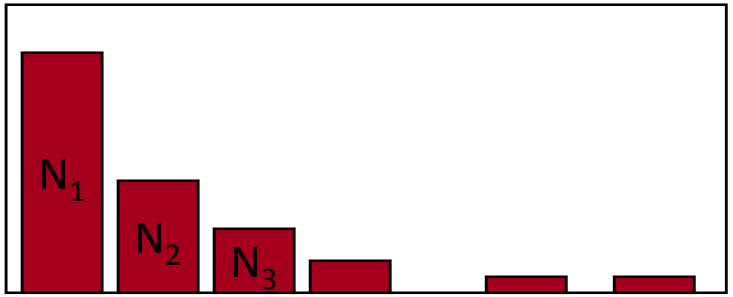
$$k^* = \frac{(k+1)N_{k+1}}{N_k}$$

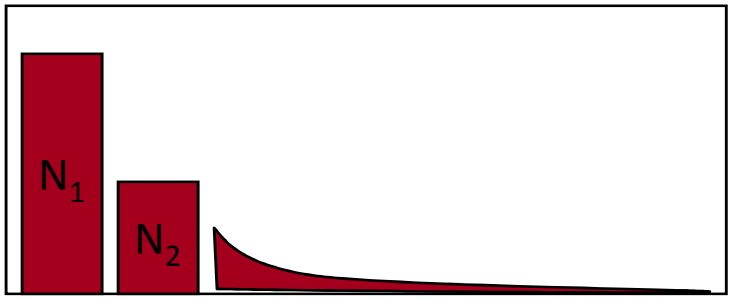


Complicações de Good-Turing

(slide de Dan Klein)

- Problema:
  - A equação $k^* = \frac{(k+1)N_{k+1}}{N_k}$ é indefinida para N_k= zero
  - Simple Good-Turing [Gale e Sampson]: substituir N_k empiricamente com uma lei de potência de melhor ajuste, uma vez que contagens se tornaram não confiáveis.
  - Interporla N_k em função de (N_{k+1}, N_{K-1})





Números Resultantes de Good-Turing

- Números por Church e Gale (1991)
- 22 milhões de palavras do AP Newswire

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

Contagem c	Good Turing c*
0	.0000270
1	0.446
2	1.26
3	2.24
4	3.24
5	4.22
6	5.19
7	6.21
8	7.24
9	8.25

Modelagem de Linguagens

Avançado:

Suavização Good-Turing

Modelagem de Linguagens

Avançado:

Suavização Kneser-Ney

Números Resultantes de Good-Turing

- Números por Church e Gale(1991)
- 22 milhões de palavras do AP Newswire

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

- Na tabela ao lado, parece que (tirando 0 e 1):

$$c^* = c - 0{,}75$$

Contagem c	Good Turing c*
0	.0000270
1	0.446
2	1.26
3	2.24
4	3.24
5	4.22
6	5.19
7	6.21
8	7.24
9	8.25

Interpolação Absoluta com Desconto

- Salve-nos algum tempo e apenas subtraia 0.75 (ou algum d)!

Bigram descontado

Peso de interpolação

$$P_{\text{AbsoluteDiscounting}}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w)$$

unigram

  - (Talvez mantendo alguns valores extras de d para contagens 1 e 2)
- Mas devemos realmente só usar o unigram regular P(w)?

Suavização I Kneser-Ney

- Melhor estimador para probabilidades de unigrams de baixa ordem!
  - Shannon game: *I can't see without my reading* ___ ?
  - "Francisco" é mais comum que "glasses"
  - … mas "Francisco" sempre vem depois de "San"
- O unigram é útil exatamente quando nós ainda não vimos esse bigram!
- Ao invés de P(w): "O quão provável é w"
- $P_{continuation}$(w): "O quão provável é w aparecer em um novo contexto?"
  - Para cada palavra, contar o número de tipos de bigram que ela completa
  - Cada tipo de bigram era um novo contexto* na primeira vez que foi visto

$$P_{CONTINUATION}(w) \propto \left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|$$

contexto = continuation

Sauvização II Kneser-Ney

- Quantas vezes w aparece como uma nova continuação:

$$P_{CONTINUATION}(w) \propto \left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|$$

- Normalizado pelo número total de tipos de bigram de palavra

$$\left|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}\right|$$

$$P_{CONTINUATION}(w) = \frac{\left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|}{\left|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}\right|}$$

Suavização III Kneser-Ney

- Metáfora Alternativa: O número de tipos de palavras vistas precedendo w

$$|\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

- normalizado pelo numero de palavras precedendo todas as palavras:

$$P_{CONTINUATION}(w) = \frac{\left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|}{\sum_{w'} \left|\{w'_{i-1} : c(w'_{i-1}, w') > 0\}\right|}$$

- Uma palavra frequente (Francisco) ocorrendo em um único contexto (San) vai ter uma probabilidade baixa de continuação

Suavização IV Kneser-Ney

$$P_{KN}(w_i \mid w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1}) P_{CONTINUATION}(w_i)$$

λ é uma constante de normalização; a massa de probabilidade que nós descontamos

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})} \left| \{w : c(w_{i-1}, w) > 0\} \right|$$

O desconto normalizado

Número de tipos de palavra que seguem w_{i-1}
= # de tipos de palavras que descontamos
= # de vezes que aplicamos desconto normalizado

Modelagem de Linguagens

Avançado:

Suavização Kneser-Ney