Introdução ao Modelo de Linguagem n-gram

Prof. Dr. Thiago de Paulo Faleiros

Roteiro da apresentação

Predição de palavras

- A água do mar é ...
  - vermelha
  - refrigerada
  - azul
  - limpa
  - aquela

Modelo de Linguagem

Sistema que pode predizer próximas palavras

- Pode atribuir a probabilidade para cada potencial próxima palavra
- Pode atribuir uma probabilidade para toda a sentença

Por que prever palavra?

Pode ser muito útil numa tarefa

- Corretor gramatical
- Reconhecimento de fala
- Geração automática de texto
- Tradução automática

Por que prever palavra?

É assim que os LLM's (*Large Language Models*) funcionam!

- LLM's são treinados para prever a próxima palavra
  - Da esquerda para a direita (autoregressivo) os LM's aprendem a prever a próxima palavra
- LLM's geram texto prevendo palavras
  - Prever a próxima palavra sequencialmente

Modelos de Linguagem formalmente

Objetivo: Computar a probabilidade de uma sentença ou uma sequência de palvras W:

$$P(W) = P(w_1, w_2, w_3, \ldots, w_n)$$

Tarefas relacionadas: probabilidade da próxima palavra:

$$P(w_n|w_1, w_2, w_3, w_4, \ldots, w_{n-1})$$

Todos os LM's computam algum desses:

$$P(W) \text{ ou } P(w_n|w_1, w_2, w_3, w_4, \ldots, w_{n-1})$$

Como estimar essas probabilidades?

Podemos apenas contar e dividir?

$$P(\text{azul}|\text{A água do mar é }) = \frac{C(\text{A água do mar é azul})}{C(\text{A água do mar é})}$$

Não! Existem muitas possibilidades
Nunca teremos dados suficientes para essa estimativa

Como computar $P(W)$ ou $P(w_n|w_1, \ldots, w_{n-1})$

Como computar a probabilidade conjunto $P(W)$:

$$P(\text{A, água, do, mar, é, azul})$$

Intuição: Vamos aplicar a regra da cadeia!

Relembrando: A regra da cadeia

Relembrando a definição de probabilidade condicional

$P(B|A) = P(A,B)/P(A)$ Reescrevendo: $P(A,B) = P(A)P(B|A)$

Mais variáveis:

$$P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$$

Regra da cadeia:

$$P(x_1, x_2, x_3, \ldots, x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$$

Computando a probabilidade conjunta das palavras de uma sentença aplicando a regra da cadeia

$$P(w_{1:n}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2}) \dots P(w_n|w_{1:n-1})$$
$$= \prod_{k=1}^{n} P(w_k|w_{1:k-1})$$

$$P(\text{“A água do mar”}) =$$
$$P(\text{A}) \times P(\text{água}|\text{A}) \times P(\text{do}|\text{A água}) \times P(\text{mar}|\text{ A água do})$$

Suposição de Markov

Simplificação:

$$P(\text{azul}|\text{A água do mar é}) \approx P(\text{azul} \mid \text{é})$$

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1})$$

Bigrama de Markov

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

Mais geralmente, nós aproximamos cada componente no produto

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1})$$

Problemas com modelo n-gram

- n-gram não trata dependências de longas distâncias
- n-gram não é bom para modelar novas sequências com significado semelhante

A solução: LLM

- Pode tratar de contextos maiores
- Por usar espaço de imersão de palavras, usa melhor os sinônimos e gera melhor novas palavras

Então por que estudar n-gram?

Um paradigma fácil e simples para introduzir várias questões dos LLM's

- Treinamento e teste
- Métrica de perplexidade
- Amostragem para gerar sentenças
- Ideias como interpolação e recuo

Estimando probabilidade dos bigramas

Estimativa de máxima verossimilhança

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)}$$

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

Um Exemplo

- $< s >$ I am Sam $< /s >$
- $< s >$ Sam I am $</s>$
- $< s >$ I do not like green eggs and ham $< /s >$

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}$$

$$P(I| < s >) = \frac{2}{3} = 0.67 \quad P(Sam| < s >) = \frac{1}{3} = 0.33 \quad P(am|I) = \frac{2}{3} = 0.67$$

$$P(< /s > |Sam) = \frac{1}{2} = 0.5 \quad P(Sam|am) = \frac{1}{2} = 0.5 \quad P(do|I) = \frac{1}{3} = 0.33$$

Mais Exemplos

Berkeley Restaurant Project Sentences

- can you tell me about any good cantonese restaurants close by
- tell me about chez panisse
- i'm looking for a good place to eat breakfast
- when is caffe venezia open during the day

Contagem de Bigramas

Mais de 9222 sentenças

	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

Probabilidades dos Bigramas

Normaliza por unigramas

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Resultado:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Probabilidade da sentença por estimativas

$$
\begin{aligned}
P(\text{<s> I want english food </s>}) &= P(I|<s>) \\
&\times P(want|I) \\
&\times P(english|want) \\
&\times P(food|english) \\
&\times P(</s>|food) \\
&= 0.000031
\end{aligned}
$$

Qual tipo de conhecimento os ngramas representam?

$$P(english|want) = .0011$$
$$P(chinese|want) = .0065$$
$$P(to|want) = .66$$
$$P(eat|to) = .28$$
$$P(food|to) = 0$$
$$P(want|spend) = 0$$
$$P(I| < s >) = .25$$

Tratando grande escala em ngramas

Probabilidades em Modelos de Linguagem são armazenadas e computadas no espaço logarítmico.
Isso evita *underflow* causado pela multiplicação de vários números pequenos

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

Se for necessário o valor de probabilidade basta aplicar exp

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

Ngramas maiores

4-grams, 5-grams

Grandes conjuntos de dados com n-gramas maiores foram liberados

- n-grams do dataset COCA (*Corpus of Comtemporary American English*) – 1 bilhão de palavras (Davies 2020)
- Google Web 5-grams (Franz and Brants 2006) – 1 trilhão de palavras
- **Eficiência:** probabilidade quantizada para 4-8 bits em vez de float com 8-bytes

Novos Modelos:

- infini-grams (Liu et al 2024)
- Nenhuma pré-computação! Em vez disso, são armazenados 5 trilhões de palavras de textos da web em um array de sufixo. Isso pode computar a probabilidade de n-gramas para qualquer valor de n!

Como avaliar modelos n-gramas?

Avaliação Extrínsica

Comparar modelo A com B

1. Coloque cada modelo em uma tarefa real
   - Tradução de Máquina, reconhecimento de fala, etc.
2. Execute a tarefa, pegue o score para A e para B
   - Quantas palavras foram traduzidas corretamente
   - Quantas palavras foram transcritas corretamente
3. Compare a acurácia para A e B

Avaliação Intrínsica

Avaliação Extrínsica nem sempre é possível

- Consome muito tempo
- Nem sempre generalizável para outras aplicações

Avaliação Intrínsica: perplexidade

- Diretamente mede o desempenho do modelo de linguagem na predição de palavras
- Não necessariamente corresponde com o desempenho da aplicação real
- Mas nos dá uma métrica geral para modelos de linguagem
- Útil para LLM's assim como para n-gramas

Conjuntos de treino e teste

Treinamos parâmetros do nosso modelo no conjunto de treinamento
Testamos o desempenho do modelo nos dados que ainda não vimos

- Um conjunto de teste é um dado ainda não visto; diferente do conjunto de treinamento
  - Intuição: queremos medir a generalização para dados desconhecidos
- Uma métrica de avaliação (como a perplexidade nos diz o quão bom nosso modelo se aplica no conjunto de teste

Escolhendo o conjunto de treino e teste

- Se nós estamos construindo um LM para uma tarefa específica
  - O conjunto de teste deve refletir a tarefa que queremos usar para o modelo
- Se estamos construindo um modelo de propósito geral
  - Precisamos de vários e diferentes conjuntos de treinamento
  - Não queremos um conjunto de treinamento ou de teste para apenas um domínio, autor ou linguagem

Treinamento no conjunto de teste

Não podemos permitir sentenças de teste no conjunto de treinamento

- Se não o LM irá atribuir para aquela sentença uma probabilidade alta de forma artificial quando for visto no teste
- Assim, atribuir um alto valor de probabilidade nos testes
- Fazendo o LM melhor do que realmente é

Isso é chamado "Treinando no conjunto de teste"
Bad science!!

Dev sets

- Se nós testarmos no conjunto de teste várias vezes implicitamente iremos ajustar as caracterísitcas do conjunto de teste
  - Noticiando que mudanças faz o modelo melhorar
- Assim, rodamos no conjunto de teste apenas uma vez, ou poucas vezes.
- Isso significa que precisamos de um terceiro conjunto de dados:
  - O conjunto de validação, ou devset, ou validation set
  - Testamos nosso LM no devset do começo ao fim
  - E então testamos nosso LM no conjunto de teste apenas uma vez

Intuição da perplexidade como métrica de avaliação: O quão bom é nosso LM?

Intuição: Um bom LM prefere sentenças "reais"

- Atribua alta probabilidade para sentença "real" ou "observada frequentemente"
- Atribua baixa probabilidade para palavras misturadas ou sentenças observadas raramente.

Intuição da Perplexidade: Predizendo próximas palavras

The Shannon Game: **Quão bem podemos prever próximas palavras?**
Unigramas são terríveis nesse jogo (Why?)
Um bom LM é aquele que atribui uma alta probabilidade para a próxima palavra que realmente ocorre

Intuição da perplexidade: O melhor LM é aquele que melhor prediz todo o conjunto de teste

- Dizemos: um bom LM é aquele que atribui uma alta probabilidade para a próxima palavra que realmente acontece
- Vamos generalizar para todas as palavras!
  - O melhor LM atribui alta probabilidade para todo o conjunto de teste
- Quando comparando dois LM's, A e B
  - Computamos P_A(conj. teste) e P_B(conj. teste)
  - O melhor LM dará maior probabilidade

Intuição de Perplexidade: Use perplexidade em vez da probabilidade crua

- Probabilidades dependem do tamanho do conjunto de teste
  - O valor de probabilidade é menor para textos grandes
  - Melhor: uma métrica por palavra e normalizada pelo tamanho
- **Perplexity** é a probabilidade inversa do conjunto de teste, normalizada pelo número de palavras

$$PP(W) = P(w_1 w_2 \dots w_n)^{-\frac{1}{n}}$$
$$= \sqrt[n]{\frac{1}{P(w_1 w_2 \dots w_n)}}$$

Intuição de Perplexidade: o inverso

Perplexidade é o inverso da probabilidade do conjunto de teste, normalizado pelo número de palavras

$$PP(W) = P(w_1 w_2 \dots w_n)^{-\frac{1}{n}}$$
$$= \sqrt[n]{\frac{1}{P(w_1 w_2 \dots w_n)}}$$

(O inverso vem da definição original de perplexidade da taxa de entropia cruzada na teoria da informação)
A probabilidade está entre [0,1], perplexidade estará entre $[1, \infty]$
Minimizando a perplexidade é o mesmo que maximizar a probabilidade

Intuição de Perplexidade: n-gramas

$$PP(W) = P(w_1 w_2 \dots w_n)^{-\frac{1}{n}}$$
$$= \sqrt[n]{\frac{1}{P(w_1 w_2 \dots w_n)}}$$

Regra da cadeia:

$$PP(W) = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(w_i | w_1 w_2 \dots w_{i-1})}}$$

Bigramas:

$$PP(W) = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(w_i | w_{i-1})}}$$

Intuição de Perplexidade: fator de ramificação da média ponderada

Perplexidade é também a média ponderada da linguagem
Fator de ramificação: número de possíveis próximas palavras que podem seguir uma palavra

- Exemplo: Uma linguagem determinística L = {vermelho, azul, verde }
- Fator de ramificação = 3 (qualquer palavra pode ser seguida por vermelho, azul ou verde)

Intuição de Perplexidade

Dado um conjunto de teste T = " vermelho vermelho vermelho vermelho azul"

- Agora, assuma um LM A onde cada palavra segue qualquer outra palavra com probabilidade igual a $\frac{1}{3}$
  - $PP_A(T) =$
    $P_A(\text{vermelho vermelho vermelho vermelho azul})^{-\frac{1}{5}} =$
    $((\frac{1}{3})^5)^{-\frac{1}{5}} = (\frac{1}{3})^{-1} = 3$
- Mas agora suponha que vermelho tem uma alta probabilidade no conjunto de treinament, para algum LM B:
  $P(vermelho) = 0.8$ $P(verde) = 0.1$ e $P(azul) = 0.1$
- Esperamos que a probabilidade seja maior, e então a perplexidade menor:
  - $PP_B(T) =$
    $P_B(\text{vermelho vermelho vermelho vermelho azul})^{-\frac{1}{5}} =$
    $(0.8 \times 0.8 \times 0.8 \times 0.8 \times 0.1)^{-\frac{1}{5}} = 0.04096^{-\frac{1}{5}} = 0.527^{-1} = 1.89$

Lower Perplexity = Melhor Modelo de Linguagem

Treinando 38 milhões de palavras, testando em 1.5 milhões de palavras

	Unigram	Bigram	Trigram
Perplexidade	962	170	109

A intuição da amostragem por suavização

- Quanto temos estatísticas esparsas

$P(w|$ denied the $)$

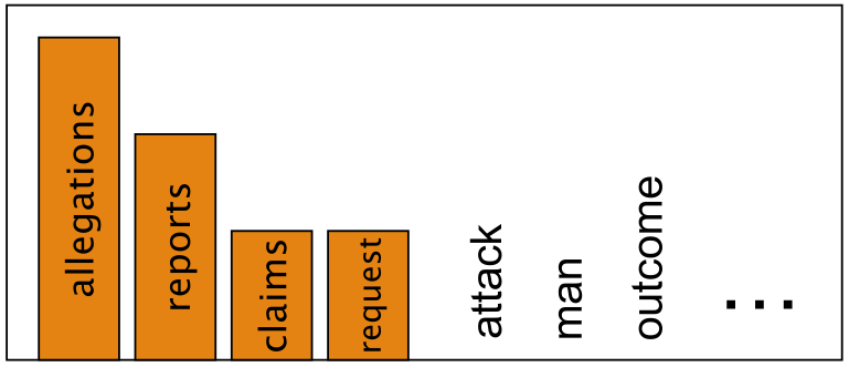
3 allegations

2 reports

1 claims

1 request

7 total



- Roubar massa de probabilidade para generalizar melhor

$P(w|$ denied the $)$

2.5 allegations

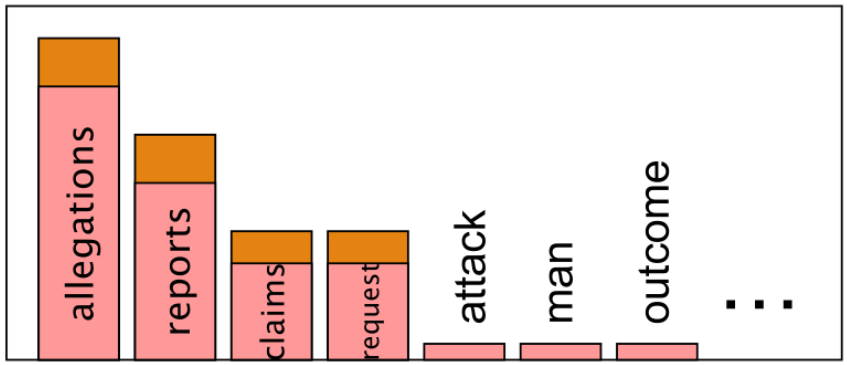
1.5 reports

0.5 claims

0.5 request

2 outros

7 total



Add-one estimation

Suavização de Laplace
Finge que vimos cada palavra uma vez mais do que foi visto
Apenas adiciona um para todas as contagens
MLE estimate:

$$P_{MLE}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

Add-1 estimate:

$$P_{Laplace}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{\sum_w(C(w_{n-1})+1)} = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

Estimador de Máxima Verossimilhança

A máxima verossimilhança estima

- alguns parâmetros do modelo M para um conjunto de treinamento T
- maximiza a probabilidade do conjunto de treinamento T dado o modelo M

Suponha a palavra “bagel” que ocorre 400 vezes em um corpus de um milhão de palavras
Qual a probabilidade que uma palavra escolhida aleatoriamente de algum outro texto será “bagel”?
MLE estimativa é 400/1000000=0.0004
Isso pode ser uma péssima estimativa para outros corpus

- Mas essa é a estimativa mais provável que “bagel” irá ocorrer 400 vezes em um milhão de palavras.

Corpus Berkeley Restaurant: 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

Laplace-smoothed bigrams

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

	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

Reconstituindo as contagens

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

	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

Compare com contagem crua dos bigramas

	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

Estimativa de Add-1 é um instrumento contudente

Então add-1 não é usado para n-gramas:

- Geralmente usamos interpolação ou recuo

Adicionar add-1 é usado para suavizar outros modelos de PLN

- Classificação de texto
- No domínio onde o número de zeros não é tão grande

Recuo e Interpolação

- Em alguns momentos é útil usar menos contexto
  - Condição para quando se tem pouco contexto
- Recuo (*Backoff*)
  - usar trigramas se não tem boa evidência
  - caso contrário usar bigrama, caso contrário usar unigrama
- Interpolação:
  - mistura de unigrama, bigrama, trigrama

Interpolação funciona melhor

Interpolação Linear

Interpolação simples:

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

$\sum_i \lambda_i = 1$
Condicionando os lambdas pelo contexto:

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

Como ajustar λ's por interpolação?

Use um corpus estendido
Escolha λ's que maximizem a probabilidade no corpus estendido:

- Ajuste a probabilidade dos n-gramas (no conjunto de treinamento)
- Busque para os valores de λ que dão as maiores probabilidades no conjunto estendido

Recuo

Suponha que vc quer:
$P(\text{panqueca}|\text{delicioso souflé})$
Se a probabilidade do trigrama é zero, use brigrama
$P(\text{panqueca}|\text{souflé})$
Se a probabilidade do bigrama é zero, use o unigrama
$P(\text{panqueca})$
Complicação: necessário descontar o n-grama de maior ordem para que a probabilidade não some maior do que 1.

Stupid Backoff

Recuo sem disconto (não um valor probabilístico)

$$S(w_i|w_{i-k+1:i-1}) = \begin{cases} \frac{c(w_{i-k+1:i})}{c(w_{i-k+1:i-1})} & \text{se } c(w_{i-k+1:i}) > 0 \\ \lambda S(w_i|w_{i-k+2:i-1}) & \text{caso contrário} \end{cases}$$

$$S(w_i) = \frac{c(w_i)}{n}$$