# Language Modeling

Luciano Barbosa

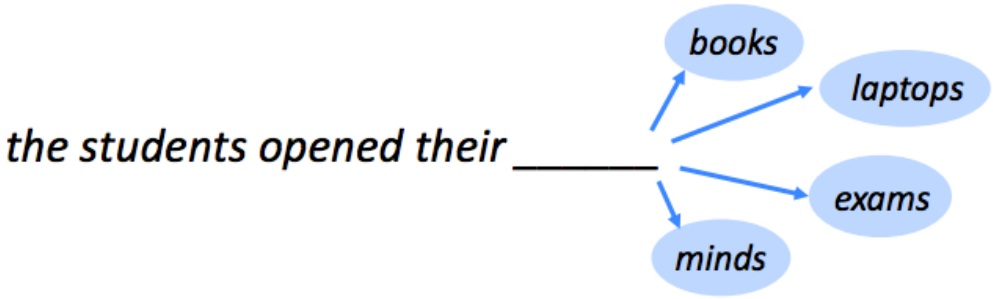
(baseado nos slides do curso de PLN de Stanford e livro Speech and Language Processing)

Centro de Informática UFPE

Universidade Federal de Pernambuco

# Definição

- Tarefa que prediz próximas palavras



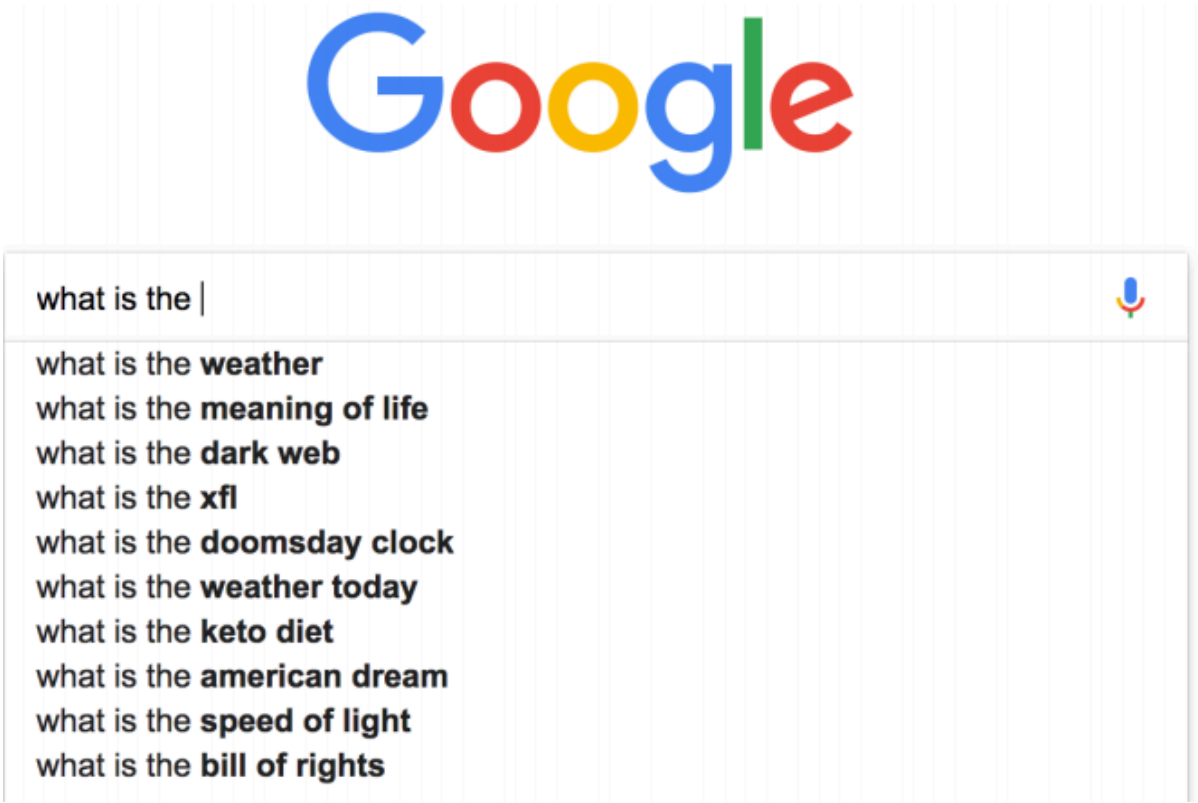

- Computar a probabilidade da próxima palavra dada uma sequência de palavras

$$P(w_5 | w_1, w_2, w_3, w_4)$$

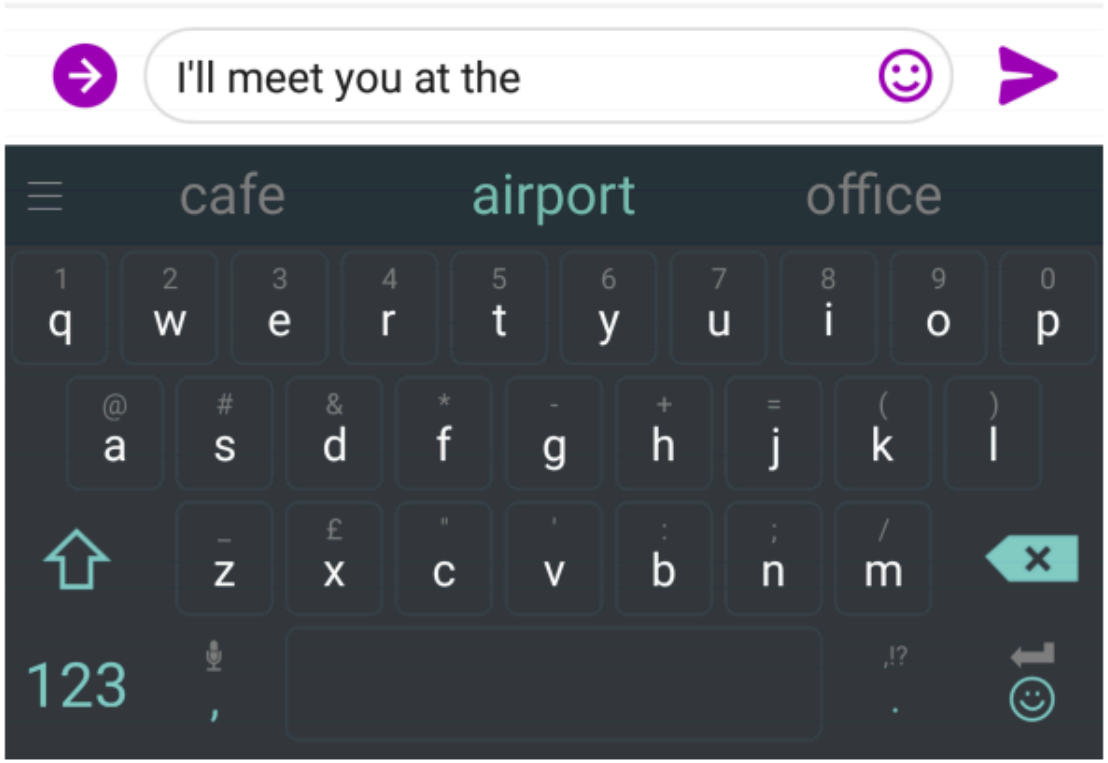
- Computar a probabilidade de uma sequência de palavras

$$P(W) = P(w_1, w_2, w_3, w_4, w_5 \ldots w_n)$$

# Onde São Usados



# Onde São Usados



# Onde São Usados

- Gerar texto ou estimar a probabilidade de um texto
- Compenente de várias tarefas de PLN
  - Corretor ortográfico
  - Reconhecedor de fala
  - Machine translation
  - Reconhecimento de escrita
  - Sumarização
  - Diálogo

# Como Computar P(W)

- Ex: P(its, water, is, so, transparent, that)
- Utiliza chain rule de probabilidade

$$\mathbf{p(B|A) = P(A,B)/P(A)} \longrightarrow \mathbf{P(A,B) = P(A)P(B|A)}$$

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

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

P("its water is so transparent") =

P(its) × P(water|its) × P(is|its water)

× P(so|its water is) × P(transparent|its water is so)

# Como Calcular as Probabilidades

- Baseado na frequência em um corpus de dados

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

- N-grams: sequência de n palavras consecutivas (trigrams, 4-grams, 5-grams etc)
- Problema: n-grams grandes são raros

# Markov Assumption

- Usadas somente as palavras mais próximas

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$

- Bigram: condicionado na palavra anterior

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

- Limitação: linguagem tem dependências de longa distância

"The computer which I had just put into the machine room on the fifth floor crashed."

# Estimando Probabilidades

- Maximum Likelihood Estimate

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

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

$$P(\texttt{I} \mid \texttt{<s>}) = \tfrac{2}{3} = .67 \qquad P(\texttt{Sam} \mid \texttt{<s>}) = \tfrac{1}{3} = .33 \qquad P(\texttt{am} \mid \texttt{I}) = \tfrac{2}{3} = .67$$

$$P(\texttt{</s>} \mid \texttt{Sam}) = \tfrac{1}{2} = 0.5 \qquad P(\texttt{Sam} \mid \texttt{am}) = \tfrac{1}{2} = .5 \qquad P(\texttt{do} \mid \texttt{I}) = \tfrac{1}{3} = .33$$

# Exemplo: Berkeley Restaurant Project sentences

can you tell me about any good cantonese restaurants close by

mid priced thai food is what i'm looking for

tell me about chez panisse

can you give me a listing of the kinds of food that are available

i'm looking for a good place to eat breakfast

when is caffe venezia open during the day

# Contagem dos Bigrams

- Total of 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 |

# Resultado

- Contagem dos Unigrams

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

- Probabilidades

| | 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 |

# Estimando a Probabilidade de uma Sentença

$$P(\text{<s> I want english food </s>}) =$$
$$P(\text{I}|\text{<s>})$$
$$\times\ P(\text{want}|\text{I})$$
$$\times\ P(\text{english}|\text{want})$$
$$\times\ P(\text{food}|\text{english})$$
$$\times\ P(\text{</s>}|\text{food})$$
$$=\ .000031$$

# Na Prática

- Cálculo em log
  - Evitar overflow
  - Adicionar é mais rápido que multiplicar

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

# Problema com Esparsidade

- OOV: out of vocabulary

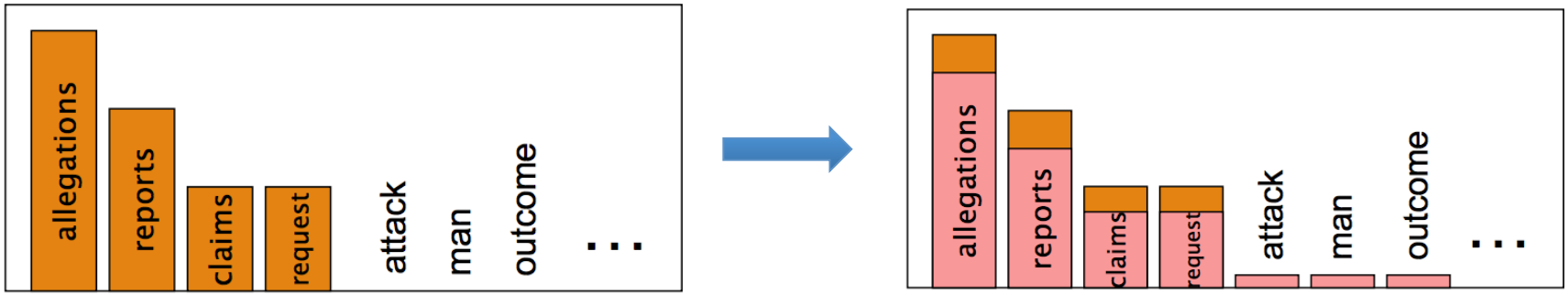
Training set:
… denied the allegations
… denied the reports
… denied the claims
… denied the request

P(“offer” | denied the) = 0

- Test set
  … denied the offer
  … denied the loan

# Smoothing

- Retira probabilidade dos n-grams no corpus



# Laplace Smoothing

- Adiciona 1 a contagens

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

# Contagem dos Bigrams com Laplace

- Total of 9222 sentenças

| | 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 |

# Contagem dos Bigrams com 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 |

# Backoff e Interpolação

- Usa menos contexto (menores n-grams)
- Backoff
  - Usar trigram se tiver dados suficientes
  - Senão, brigram ou então unigram
- Interpolação: combina unigram, bigram e trigram

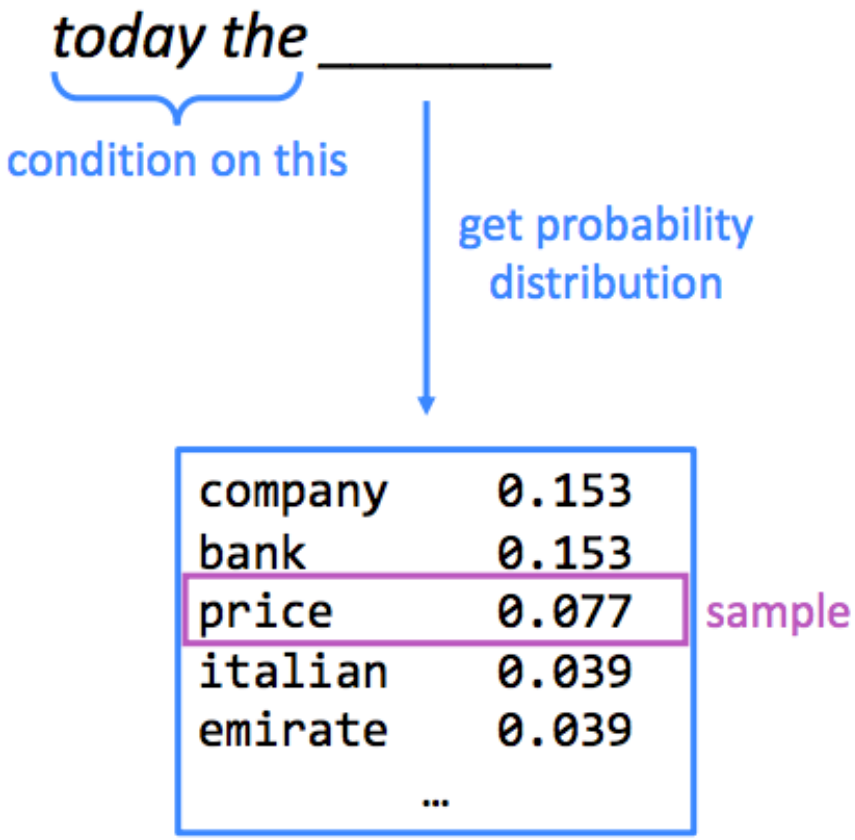
# Interpolação Linear

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

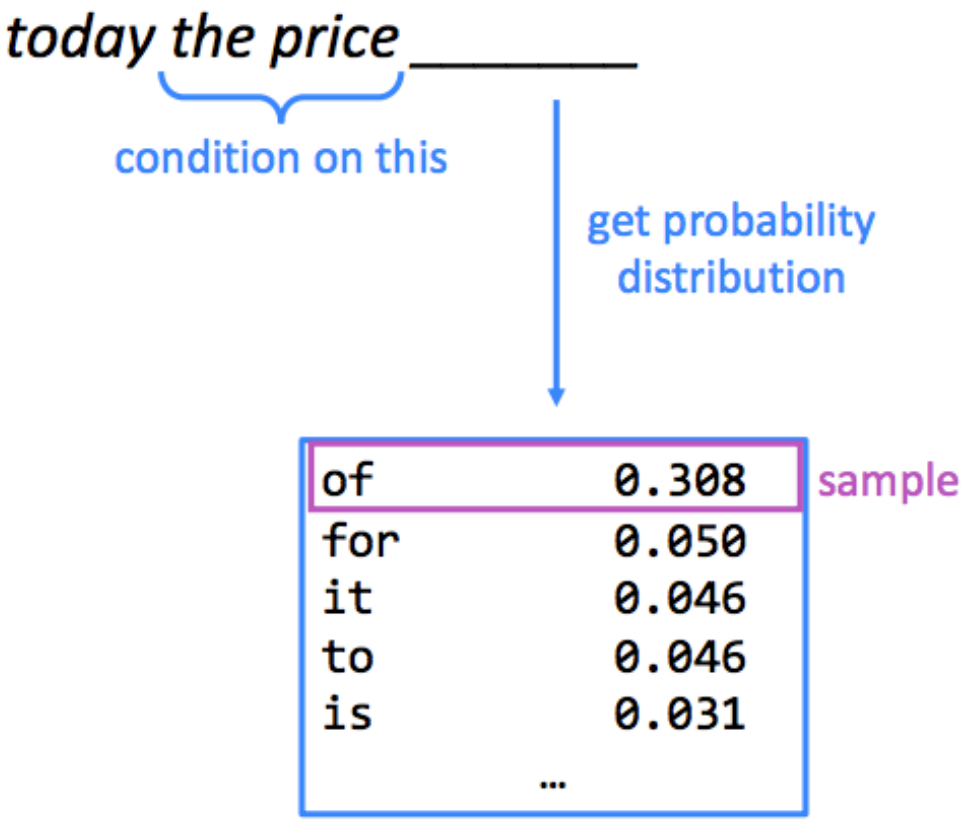
$$\sum_i \lambda_i = 1$$

- Usar um hold-out set para encontrar lambdas

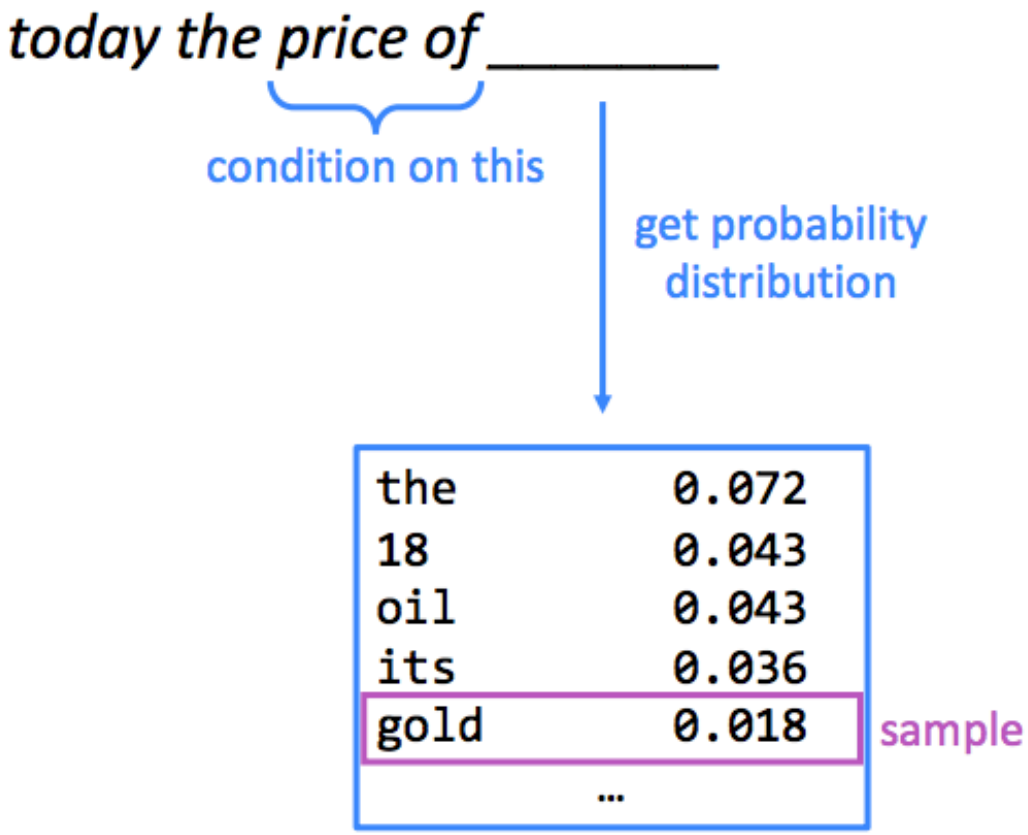
# Gerando Sentenças



# Gerando Sentenças



# Gerando Sentenças



# Limitação: Contexto Limitado

~~*as the proctor started the clock, the*~~ *students opened their* _____

discard

condition on this

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

"students opened their" occurred 1000 times

"students opened their books" occurred 400 times

- → P(books | students opened their) = 0.4

"students opened their exams" occurred 100 times

- → P(exams | students opened their) = 0.1

# Limitação: Armazenamento dos Contadores

**Storage**: Need to store count for all *n*-grams you saw in the corpus.

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

# Neural Language Models: Janela Fixa

output distribution

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

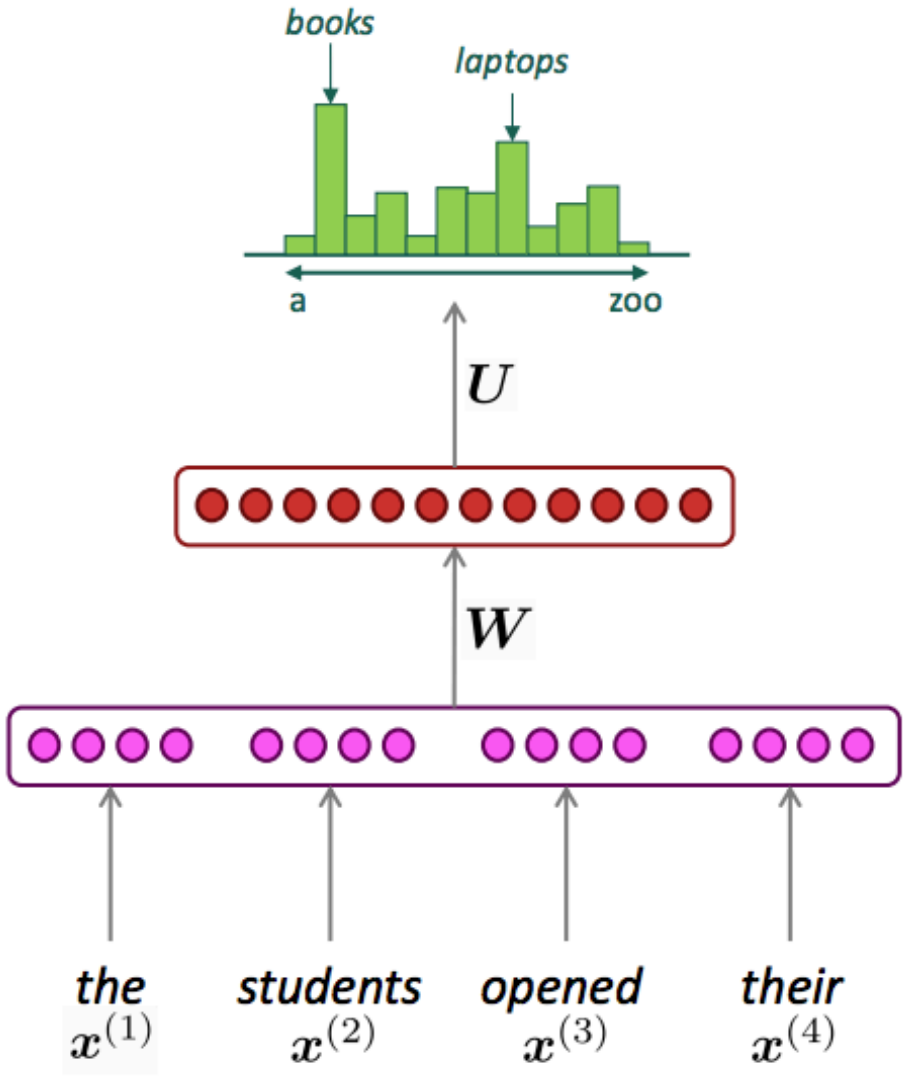
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

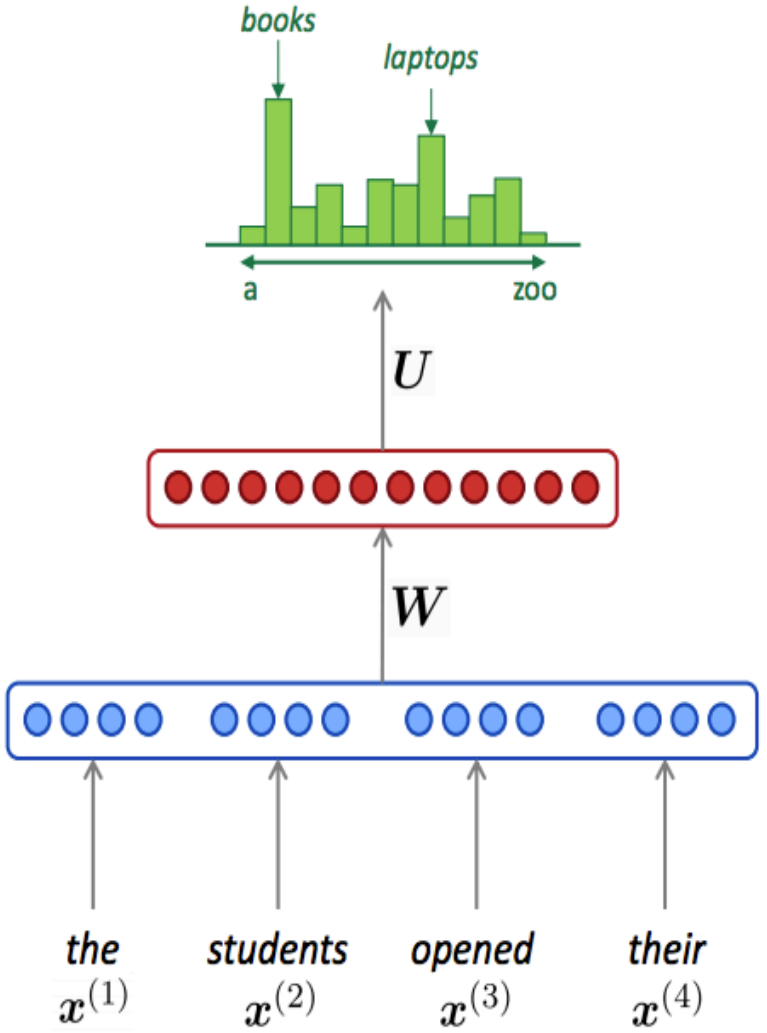
words / one-hot vectors

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$



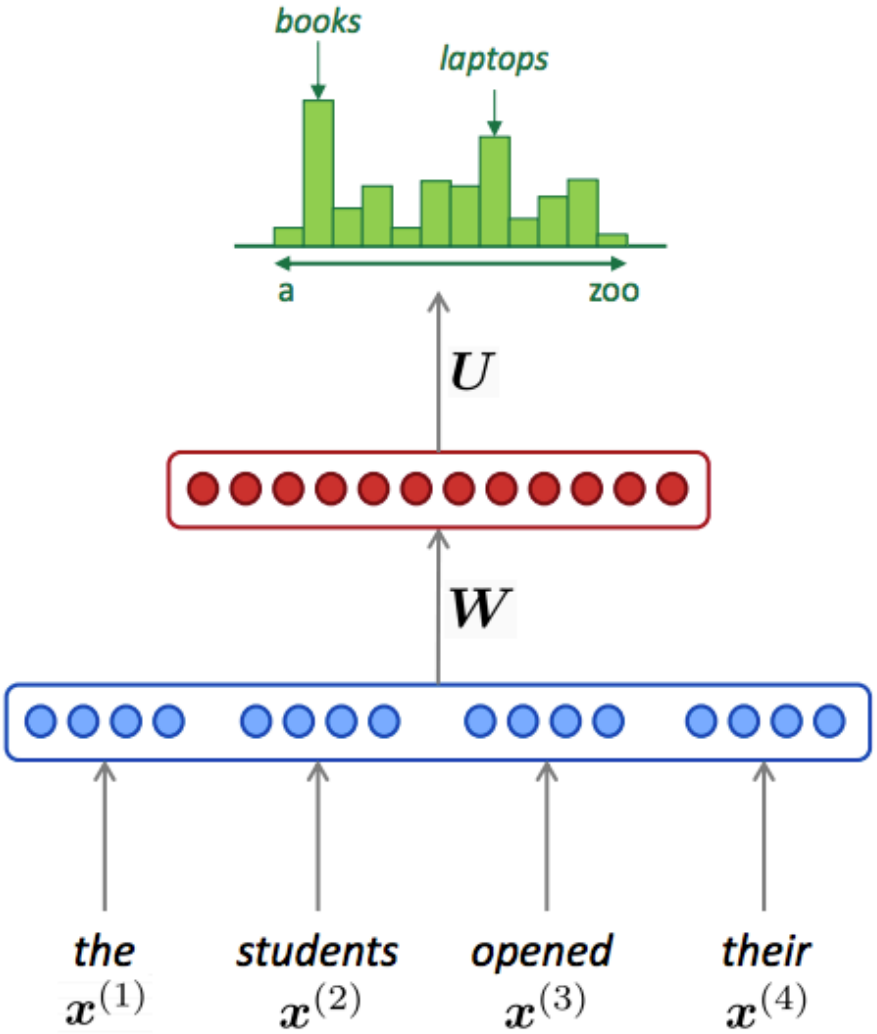
# Neural Language Models: Janela Fixa

- Vantagens:
  - Não tem problema com esparsidade
  - Não precisa armazenar os contadores dos n-grams



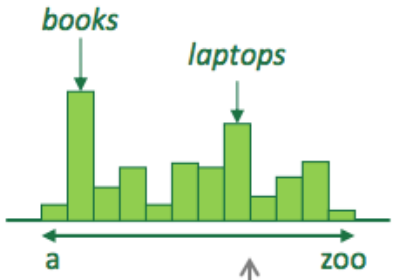
# Neural Language Models: Janela Fixa

- Desvantagens:
  - Janela fixa pequena
  - Aumento da  janela aumenta a complexidade



# Recurrent Neural Networks (RNN)

$$\hat{\boldsymbol{y}}^{(4)} = P(\boldsymbol{x}^{(5)}|\text{the students opened their})$$




**output distribution**

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

**hidden states**

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

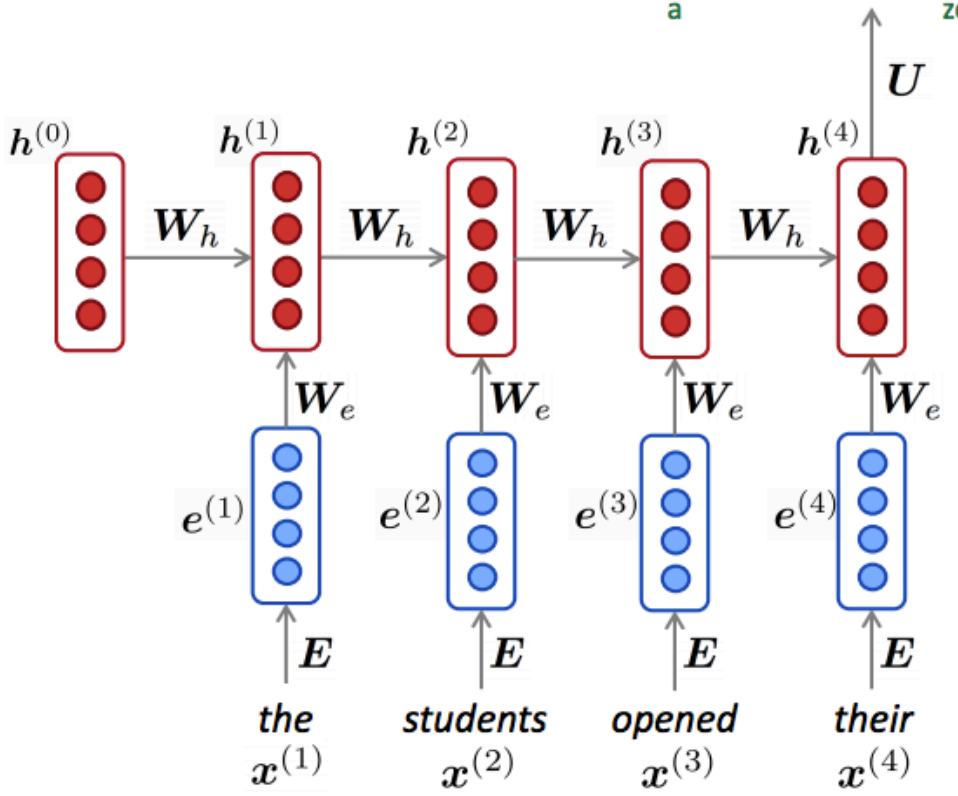
$\boldsymbol{h}^{(0)}$ is the initial hidden state

**word embeddings**

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

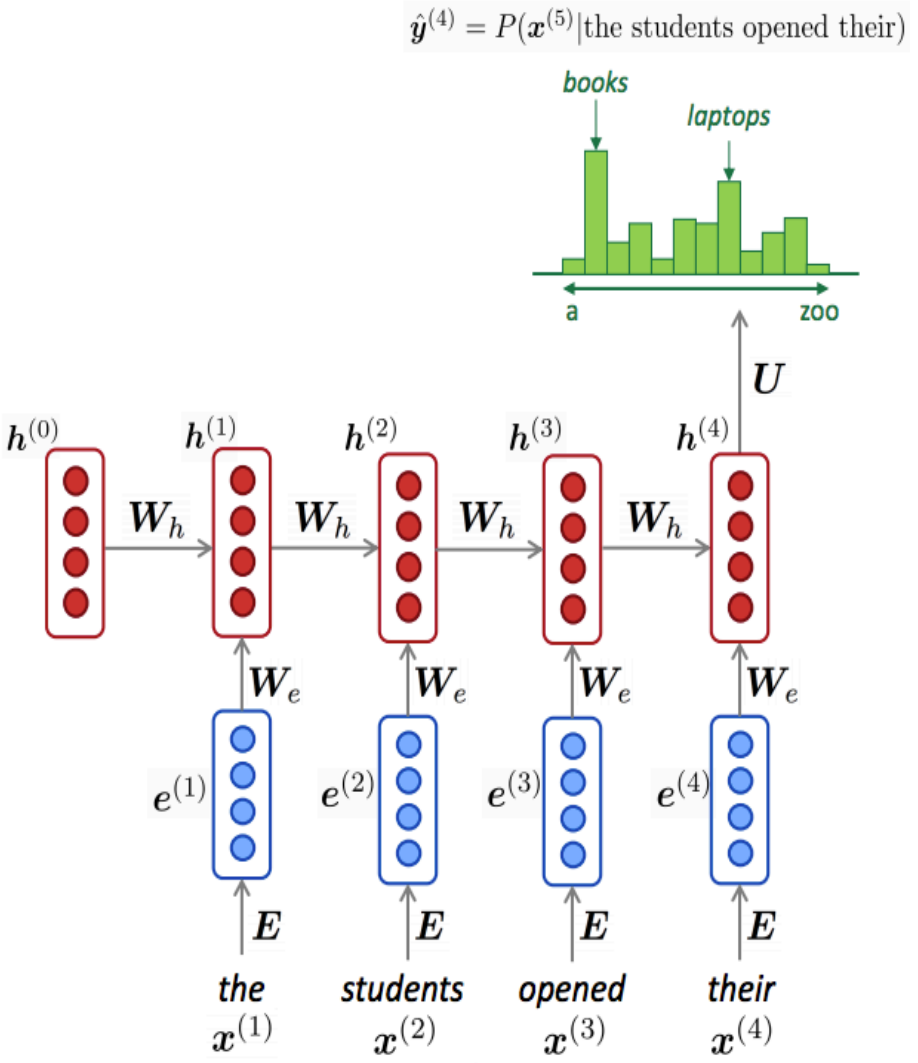
**words / one-hot vectors**

$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$

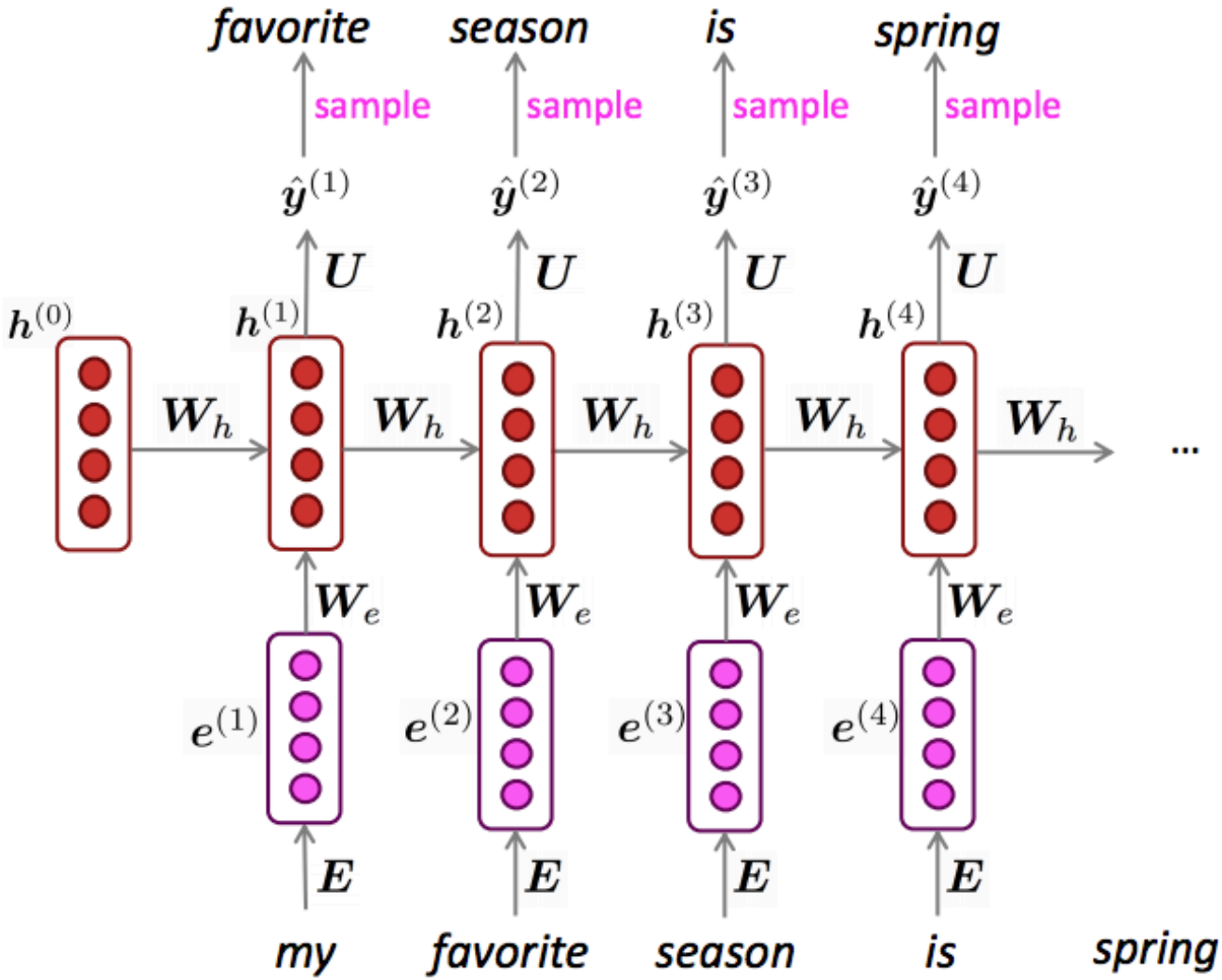


# Recurrent Neural Networks (RNN)

- Vantagens
  - Pode processar sequência de qualquer tamanho
  - Modelo não aumenta com o tamanho da sequência
  - Usa informação anterior
  - Mesmos pesos utilizados em cada passo
- Desvantagens
  - Lenta
  - Na prática, tem dificuldade em guardar informação de palavras muito anteriores



# Geração de Texto



# Exemplo de Texto Gerado com Discursos de Obama

*The United States will step up to the cost of a new challenges of the American people that will share the fact that we created the problem. They were attacked and so that they have to say that all the task of the final days of war that I will not be able to get this done.*

https://medium.com/@samim/obama-rnn-machine-generated-political-speeches-c8abd18a2ea0

# Exemplo de Texto Gerado com Para Receitas

```
      Title: CHOCOLATE RANCH BARBECUE
Categories: Game, Casseroles, Cookies, Cookies
      Yield: 6 Servings

      2 tb Parmesan cheese -- chopped
      1 c  Coconut milk
      3    Eggs, beaten

Place each pasta over layers of lumps. Shape mixture into the moderate oven and simmer
until firm. Serve hot in bodied fresh, mustard, orange and cheese.

Combine the cheese and salt together the dough in a large skillet; add the ingredients
and stir in the chocolate and pepper.
```

https://gist.github.com/nylki/1efbaa36635956d35bcc

# Avaliação de LMs: Extrínsica

- Colocar o LM em uma tarefa
  - Corretor ortográfico, reconhecedor de fala, MT
- Executar a tarefa e calcular a acurácia entre modelos
- Problema: pode ser custoso criar essas tarefas

# Avaliação de LMs: Intrínsica

- Perplexity
- Funciona bem quando o conjunto de teste é “próximo” do de treinamento
- O modelo LM é que melhor prediz um sequência de teste

# Perplexity

- Quanto menor, melhor o modelo
- Alta probabilidade -> baixo grau de surpresa

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

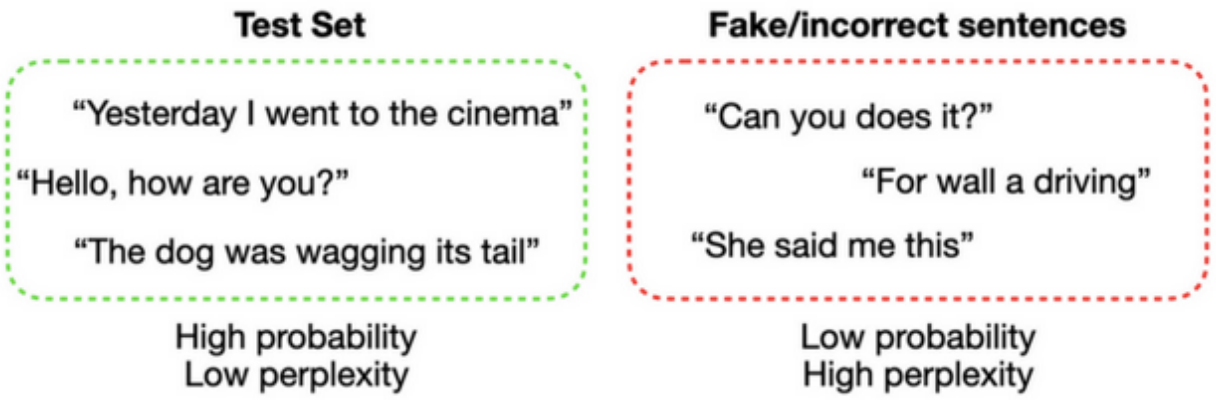
$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}$$

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

# Perplexity

- Quanto menor, melhor o modelo
- Alta probabilidade -> baixo grau de surpresa



# Comparação de Estratégias

*n*-gram model →

Increasingly complex RNNs ↓

| Model | Perplexity |
|---|---|
| Interpolated Kneser-Ney 5-gram (Chelba et al., 2013) | 67.6 |
| RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013) | 51.3 |
| RNN-2048 + BlackOut sampling (Ji et al., 2015) | 68.3 |
| Sparse Non-negative Matrix factorization (Shazeer et al., 2015) | 52.9 |
| LSTM-2048 (Jozefowicz et al., 2016) | 43.7 |
| 2-layer LSTM-8192 (Jozefowicz et al., 2016) | 30 |
| **Ours small** (LSTM-2048) | 43.9 |
| **Ours large** (2-layer LSTM-2048) | 39.8 |

↓ Perplexity improves (lower is better)

**Source:** https://research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words/