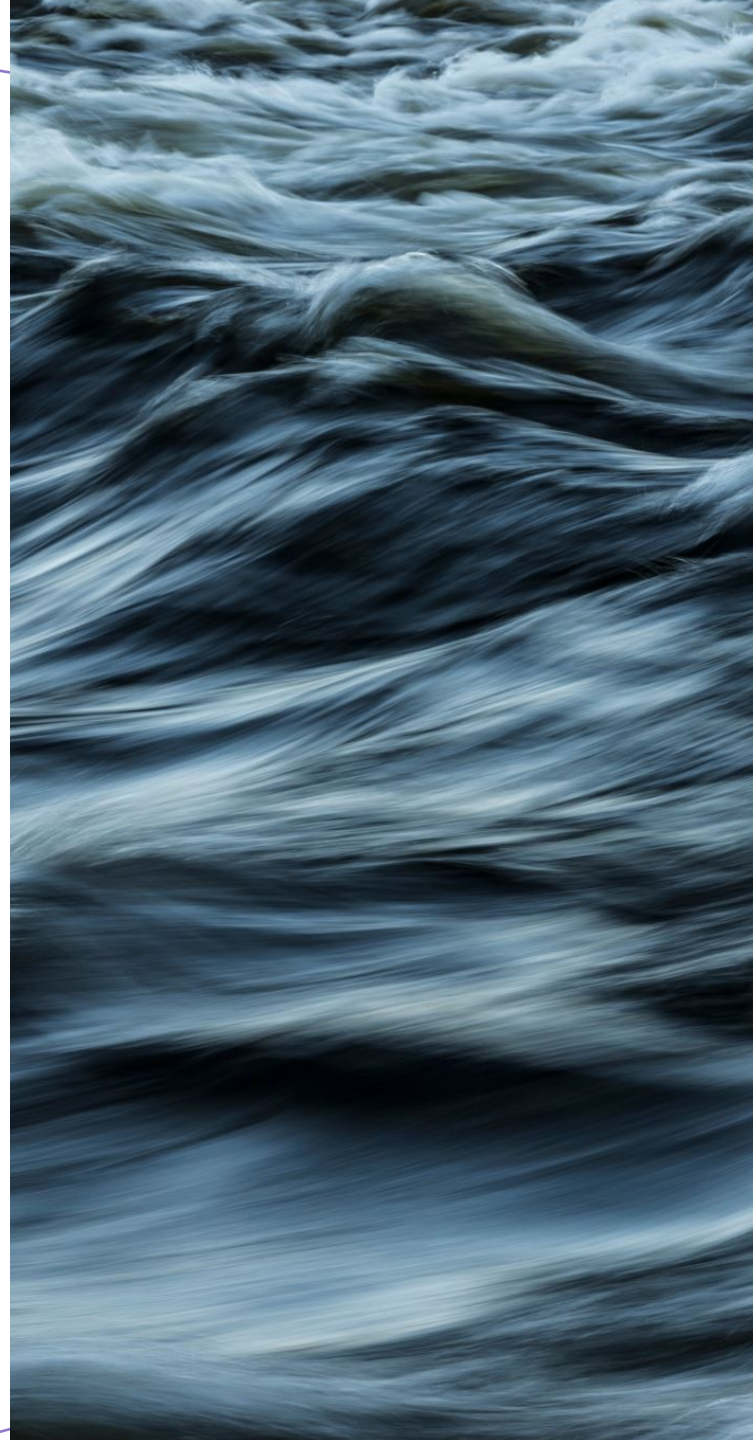
PROCESSAMENTO DE LINGUAGEM NATURAL

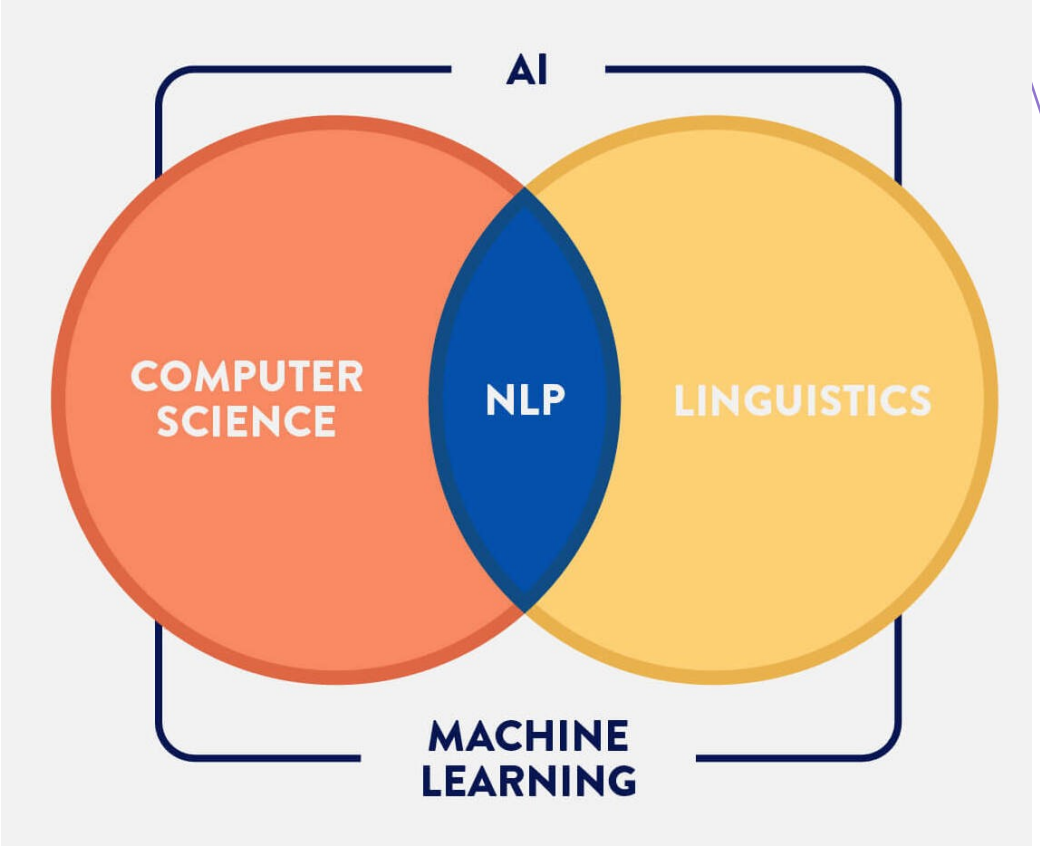
PAULO PIROZELLI

PÓS-DOUTORANDO NO C4AI-USP



PROCESSAMENTO DE LINGUAGEM NATURAL

- Campo da IA que lida com tarefas relacionadas à manipulação e compreensão da linguagem humana.
- Interface entre ciência da computação e linguística.
- Aplicações do dia a dia: editor de texto, mecanismos de busca, chatbots, reconhecimento de voz, sugestões de busca, etc.



ALGUMAS TAREFAS DE PLN

- Language modeling
- Question answering
- Summarization
- Information retrieval
- Machine translation
- Text Classification
- Sentiment analysis
- Topic modeling
- ...

ANÁLISE DE SENTIMENTO

- Determinar se um texto é positivo ou negativo (ou outra categorização).
- Ex: “O filme que acabou de estrear é péssimo!”
- Método comumente empregado: **Naive Bayes**
- Para um documento *d* e um conjunto de classes $c \in C$,

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d)$$

- Usando o teorema de Bayes, a equação se transforma em:

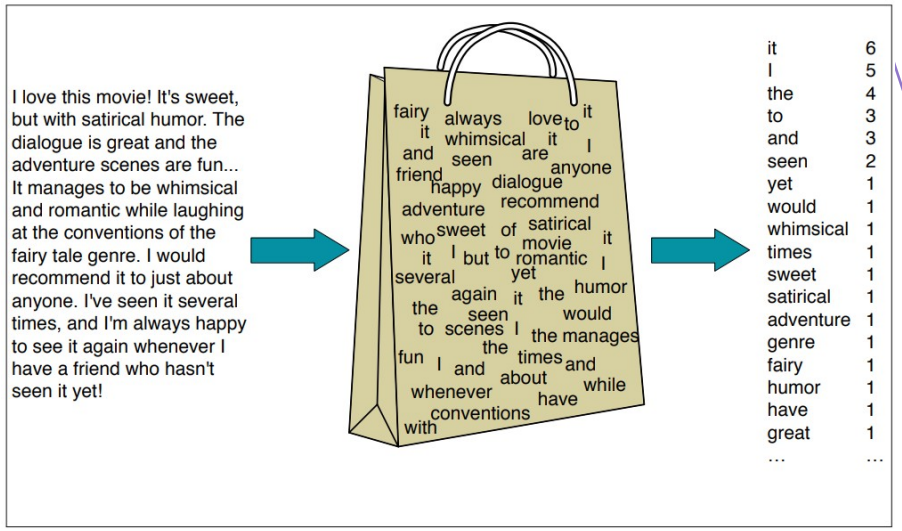
$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} \frac{P(d|c)P(c)}{P(d)}$$

- Como *P(d)* não mudar para as classes, podemos simplificar para

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} P(d|c)P(c)$$

ANÁLISE DE SENTIMENTO (CONT.)

- Representando um documento como uma série de palavras (feat
  vira: $\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(f_1, f_2, ..., f_n|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$ lihood



- $$P(f_1, f_2, ..., f_n|c) = P(f_1|c) \cdot P(f_2|c) \cdot ... \cdot P(f_n|c)$$

  elas são $c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{f \in F} P(f|c)$

ANÁLISE DE SENTIMENTO (CONT.)

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

$$P(-) = \frac{3}{5} \qquad P(+) = \frac{2}{5}$$

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \qquad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“no”}|-) = \frac{1+1}{14+20} \qquad P(\text{“no”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \qquad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

$$P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

ANÁLISE DE SENTIMENTO (CONT.)

- Para evitar underflow e acelarar a computação, fazemos uma transformação logarítimica, já que esta é uma operação monotć

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} \log P(c) + \sum_{i \in positions} \log P(w_i|c)$$

- **Binary NB**: Em classificação de texto, também é comum tratar as frequências das palavras como binárias, pois é mais relevante se uma palavra aparece ou não do que sua frequência relativa.

NAIVE BAYES

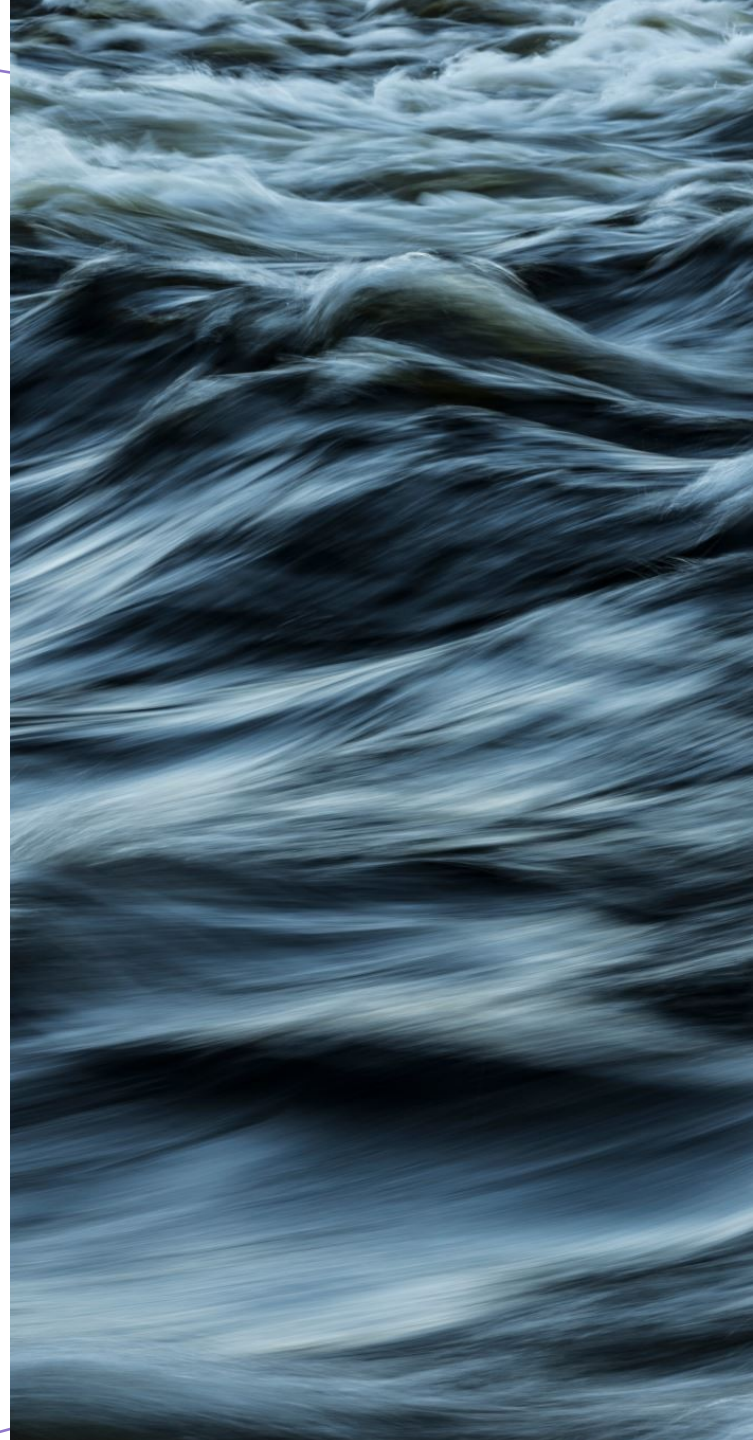
- Ignora a ordem das palavras e constituintes sintáticos.
- Não dá conta de situações ligeiramente mais complexas.
- Exemplo:

  O atleta estava feliz POSITIVO

  O jogador era infeliz NEGATIVO

  O carteiro não estava feliz POSITIVO

REDES NEURAIS



MACHINE LEARNING TRADICIONAL

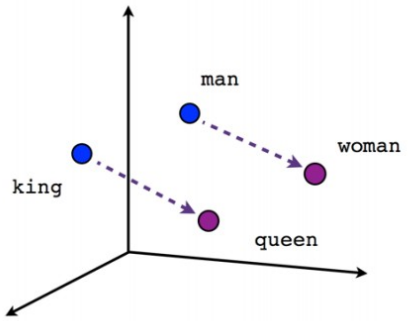
- Uso de pipelines complexos que realizam o processamento do texto
  - Normalização do texto (tokenização, steming), análise sintática, papéis semânticos, redes semânticas, traduções, etc.
- I’m running late Eu estou atrasado
- Problemas:
  - Muitas exceções existentes na língua
  - Custo de embutir conhecimento linguístico
  - Adaptação para cada língua

REDES NEURAIS

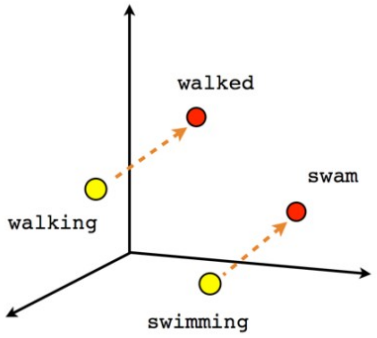
- A partir dos anos 2010, as redes neurais passaram a dominar o campo de PLN
- Possibilitado por mudanças tecnológicas:
  - Grande disponibilidade de dados com a internet
  - Maior poder computacional com o uso de GPUs
  - Frameworks especializados (Pytorch, Tensorflow, Keras,...)
- Vantagens das redes neurais:
  - Permitem construir aplicações end-to-end
  - Arquiteturas generalizáveis para várias línguas
  - Pouco conhecimento linguístico explicitamente envolvido
- Desvantagens:
  - Maior tempo de treinamento
  - Aplicações pouco transparentes

EMBEDDINGS

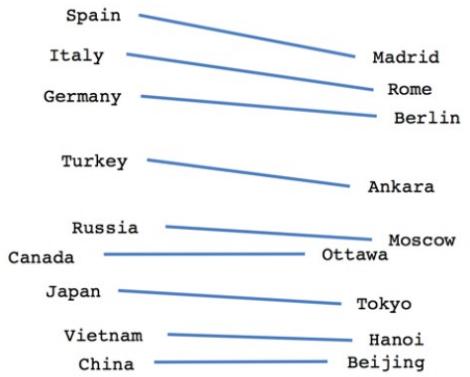
- Representações vetoriais de unidades semânticas (geralmente tokens)
  - Palavras são representadas como pontos em um espaço R^n
- Podem ser construídos sem redes neurais, mas estas proporcionam uma maneira simples e automatizada de construir tais vetores
- Permitem fazer operações matemáticas com texto
- Podem ser usados diretamente em clusterização, representações analógicas, etc.



Male-Female



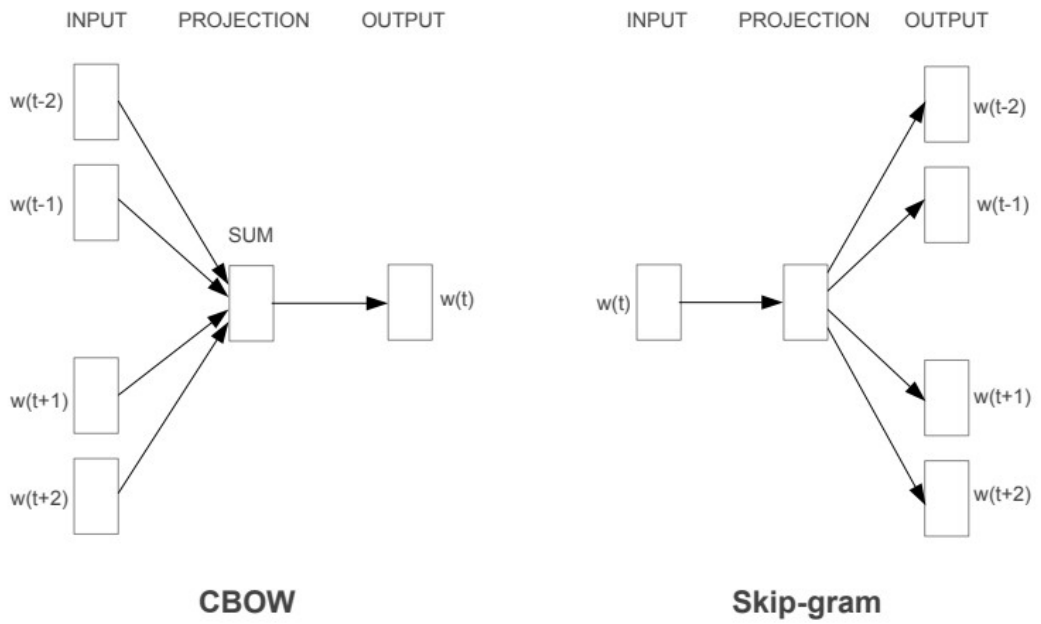
Verb tense



Country-Capital

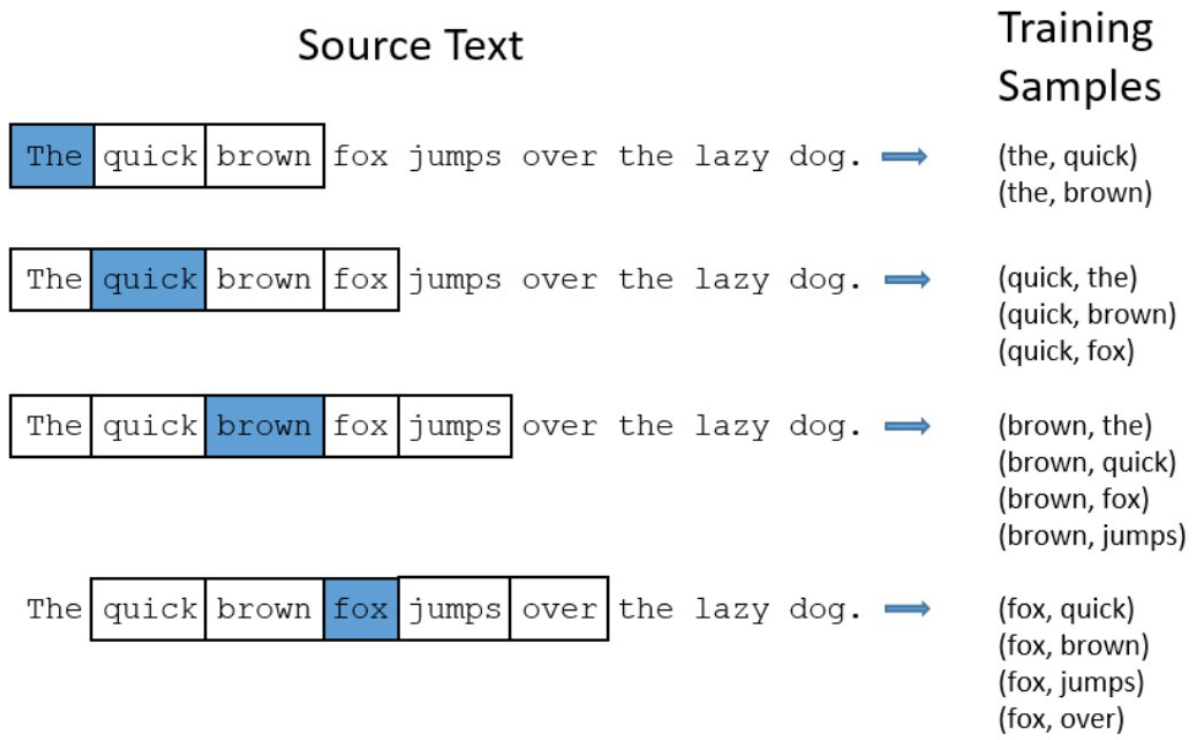
WORD2VEC

- Um das arquiteturas mais populares para produção de embeddings
- Language modeling
- Ex: qu

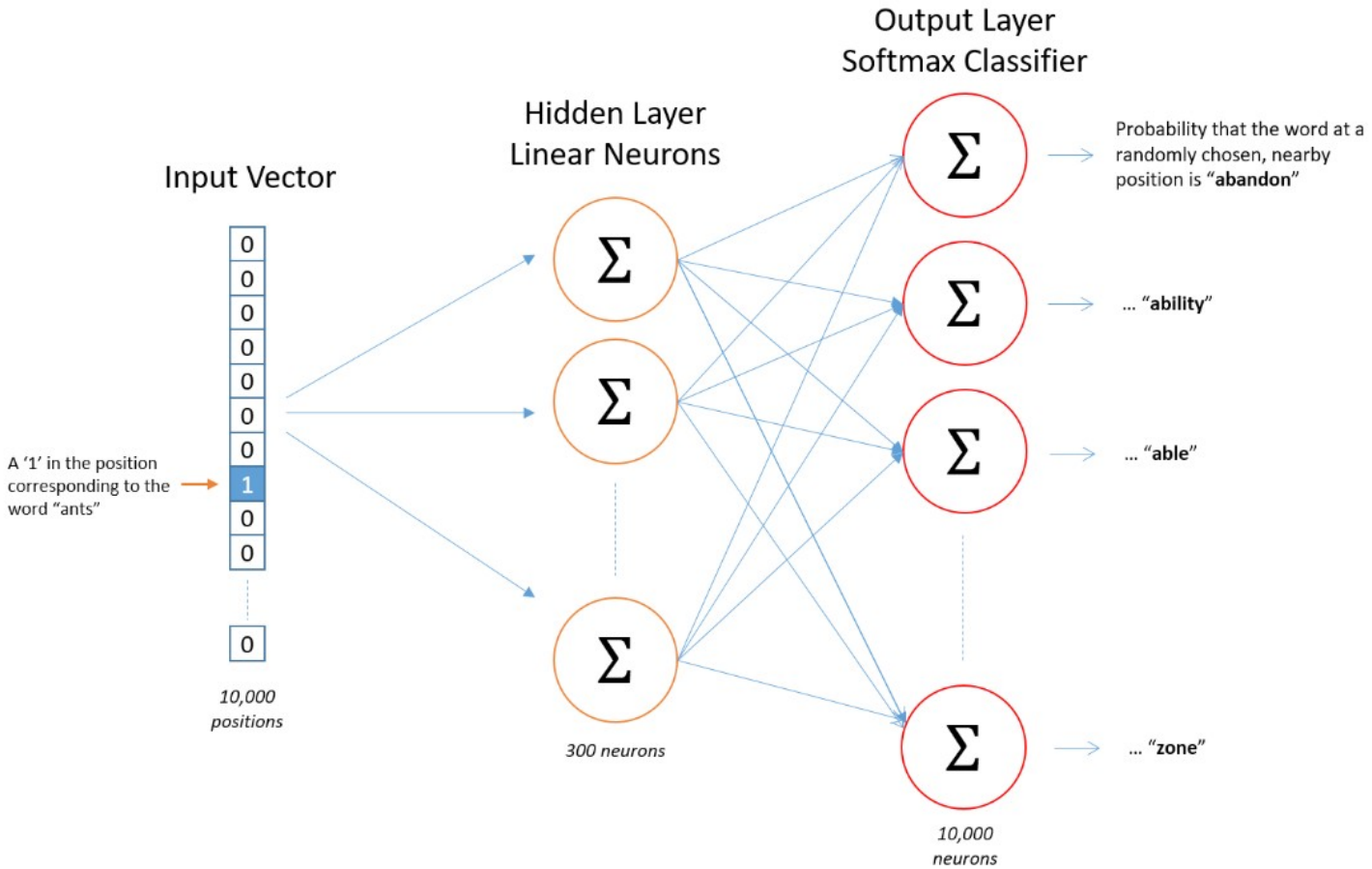


WORD2VEC (CONT.)

- Aprendizado **semi-superviosionado**: o conjunto treinamento é construído automaticamente



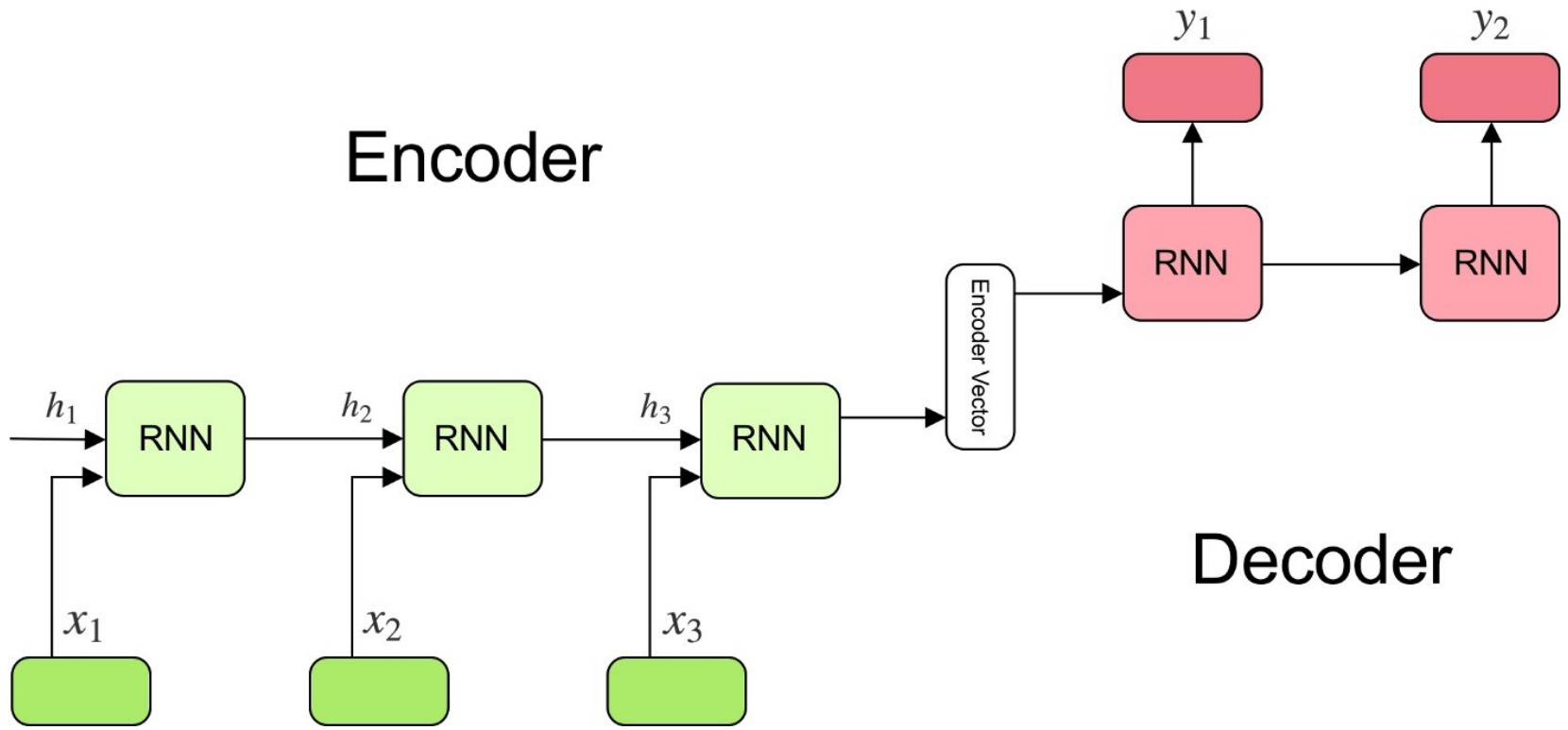
WORD2VEC (CONT.)



MODELOS TRANSFORMERS

- Tipo de redes neurais mais utilizada em PLN nos últimos anos (> 2016)
- Textos são sequências de palavras, em que a ordem dos elementos (palavras) é essencial.
- **Problema:** as sequências de entrada e saída têm tamanhos variáveis. Como lidar com isso, se uma rede neural precisa ter um tamanho fixo de entrada e saída?
- A solução tradicional é usar uma arquitetura encoder-decoder junto com uma rede neural recorrente.
- Estado da arte até os anos 2018 (p. ex., ELMo).

ENCODER-DECODER



LIMITAÇÕES DA ARQUITETURA

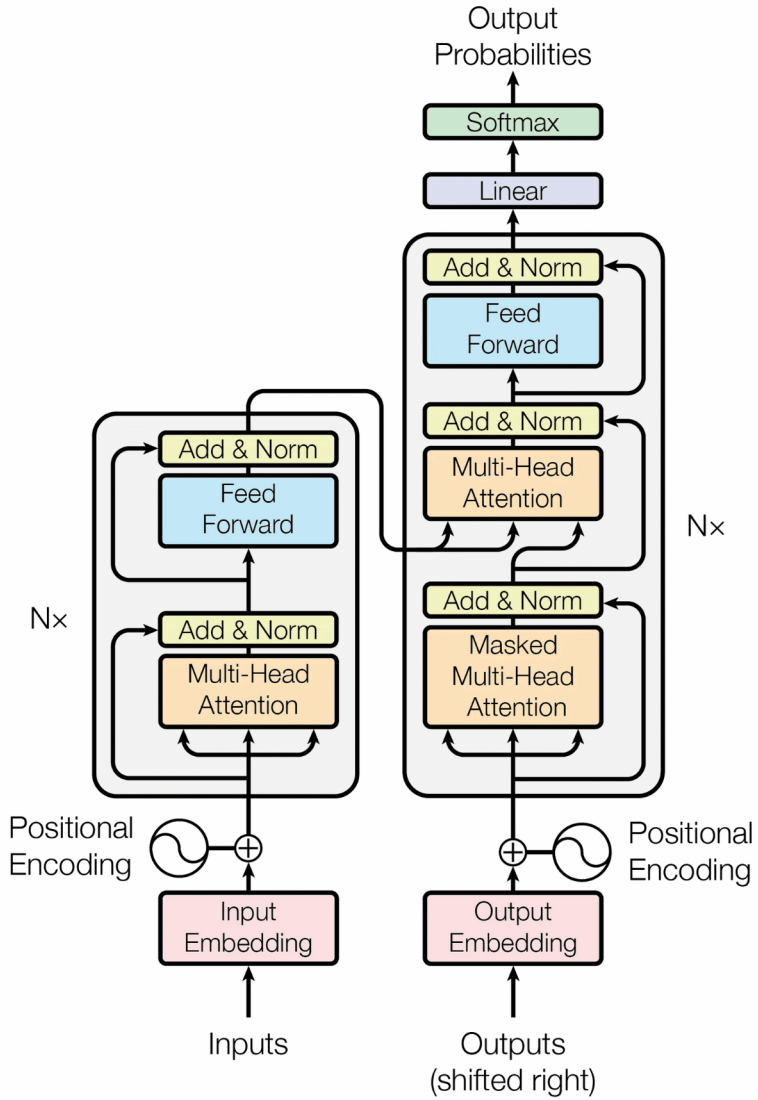
- Há várias limitações para esses modelos encoder-decoder:
  1. As computações são feitas de maneira sequencial
  2. Toda a informação sobre a entrada é armazenada em uma única representação do decoder
  3. As informações dos primeiros elementos vão se perdendo ao longo da sequência
  4. Dificuldade para treinamento, por causa do gradiente que explode ou desaparece
     - Ex: $2*10^{20}$ (exploding), $0.1*10^{20}$ (vasinhing)

TRANSFORMERS

- Transformers são uma arquitetura de redes neurais apresentadas em 2016, no artigo “Attention is all you need”.
- Basicamente, os tranformers transformam um problema sequencial em um problema não-sequencial.
- Eles fazem isso de duas maneiras:
    1. Inserindo uma informação sobre a posição na sequência de maneira explícita nos elementos – *Positional encoding*: um vetor único para cada posição da sequência *n*
    2. Usando *mecanismos de atenção*: cada unidade olha para as demais e passa uma representação composta dos termos

Vantagens dos transformers:

1. Permitem computação em paralelo.
2. Podem ser treinados em grandes quantidades de dados; ex.: o Colossal Clean Crawled Corpus (C4), usado para o treinamento do modelo T5, contém ~750GB


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

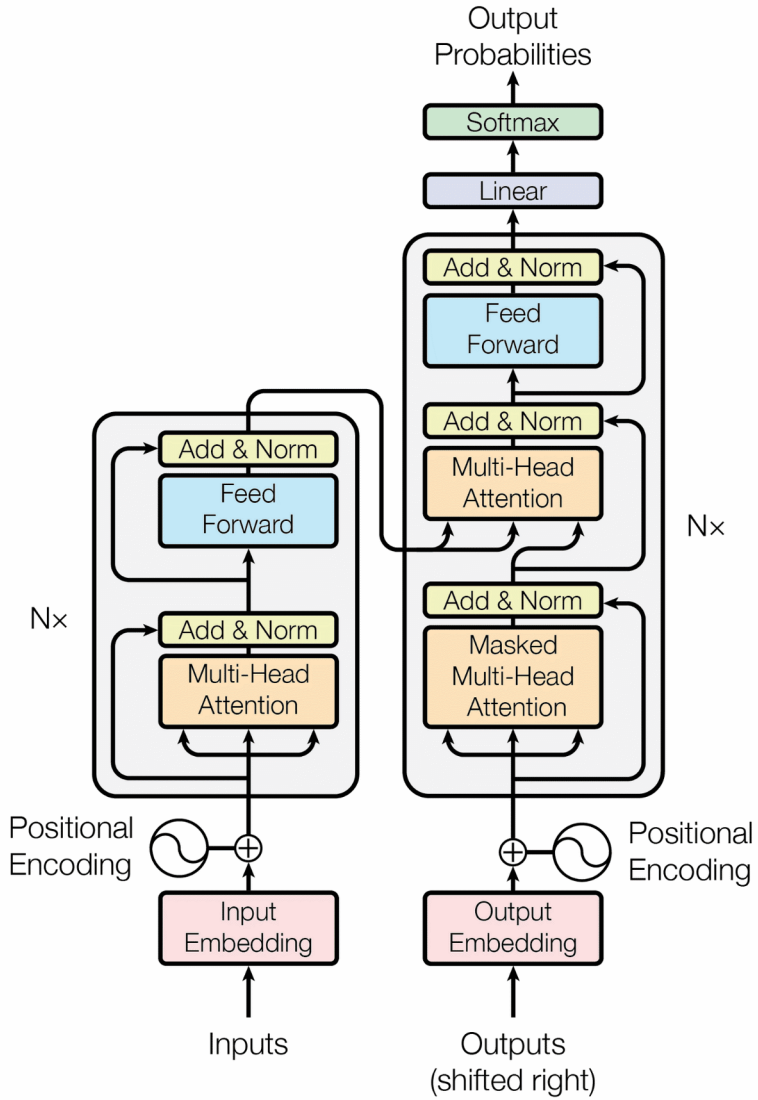
TOKENIZATION

- Processo de quebras as palavras em unidades básica: meio-termo entre a palavra (multiplicidade de formas) e o caractere (requer muita abstração)
- Lembra a noção “morfema”, as menores unidades de significado: raiz, radical, sufixo, etc.
- Exemplo: sleeping sleep + ing
- O processo de tokenização é constituído de três etapas:
  1. Quebra das palavras (processo de tokenization propriamente dito)
  2. Encoding: cada token recebe um índice único
  3. Criação de one-hot vector: um vetor em que a o valor da palavra é 1 na dimensão correspondente ao índice correspondente e 0 nas demais

TOKENIZATION (CONT.)

- Exemplo: “unstable”

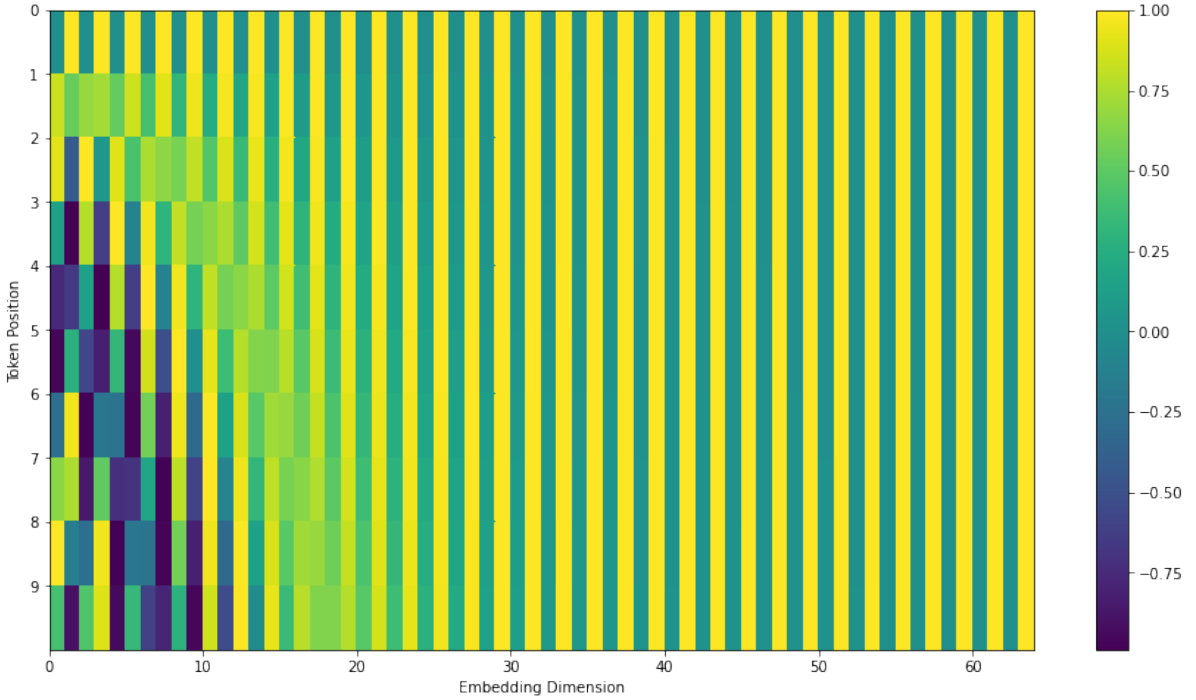
1. un + stab + le
2. {‘un’: 0, ‘stab’: 1, ‘le’: 2}
   - Esse dicionário de índices é feito com o vocabulário de todo o conjunto de dados
3. {‘un’: [1, 0, 0], ‘stab’: [0, 1, 0], ‘le’: [0, 0, 1]}


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

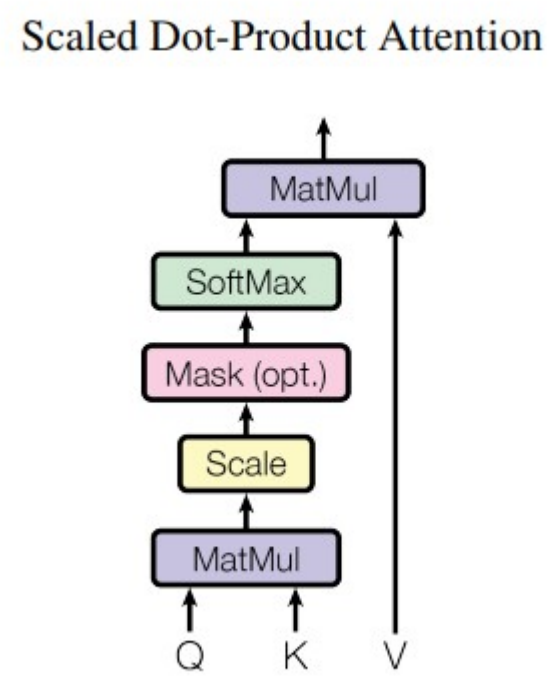
INFORMAÇÃO POSICIONAL

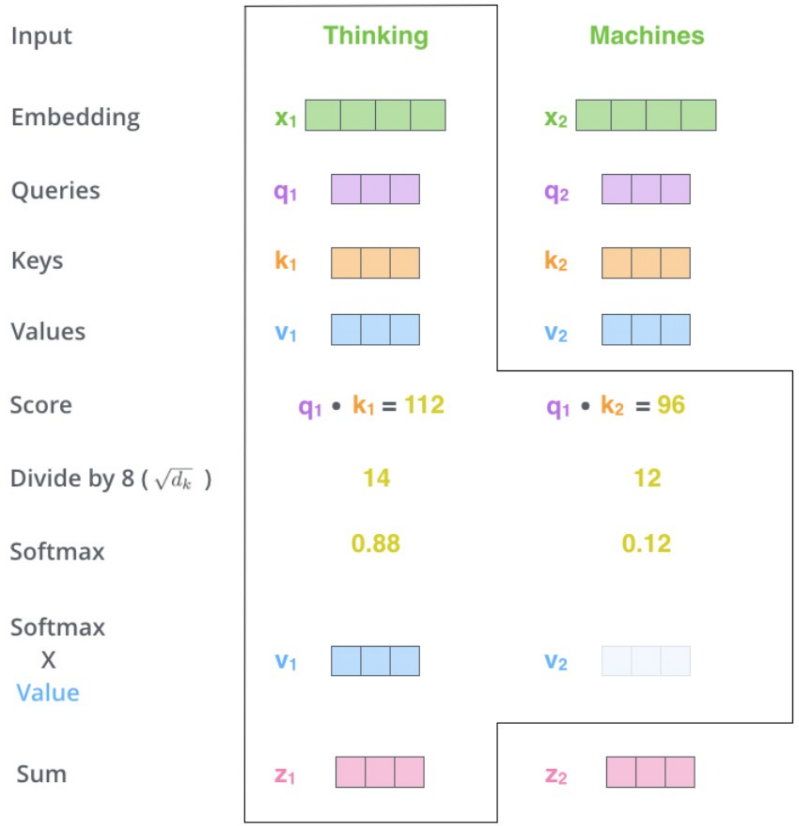
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

- Em que *pos* representa o token sob consideração e *i* representa cada uma das dimensões do vetor.
- Ex: PE(20,40) é o 40º elemento do vetor *v*, que representa o 20º token da sequência de entrada

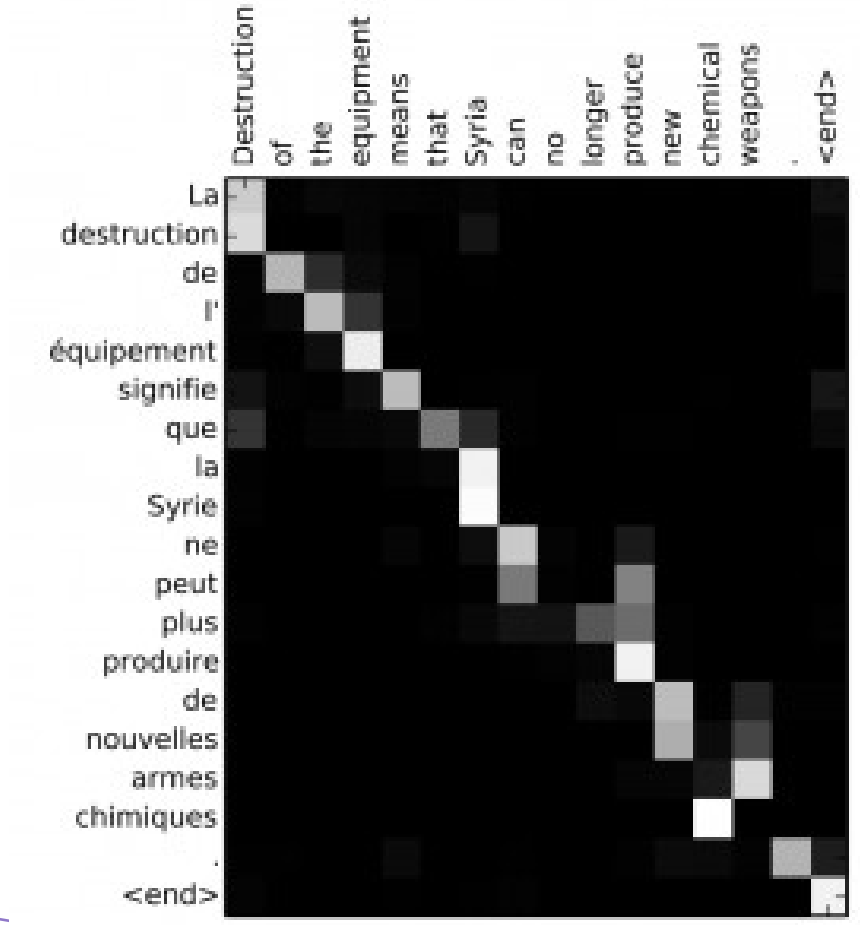


MULTI-HEAD SELF-ATTENTION MECHANISM





MULTI-HEAD SELF-ATTENTION MECHANISM (CONT.)



TREINAMENTO

- Raramente os transformers são treinados do zero. O BERTLarge, por exemplo, tem 340 milhões de parâmetros, e foi treinado em 16 Cloud TPUs (64 TPU chips no total) por 4 dias.
- No entanto, o modelo treinado pode não ser suficiente para a sua tarefa.
- Pode ser necessário treinar o modelo em novos dados.
- Além disso, a arquitetura pode não ter o tamanho correto (algo particularmente comum em tarefas de classificação)
- Em vez disso, costuma-se usar os modelos transformers de duas maneiras:
  1. Feature extraction
  2. Fine-tuning

FEATURE EXTRACTION

- O modelo transformer é congelado
- As representações geradas pelo modelo são passadas para uma nova rede, que é treinada para a tarefa em questão

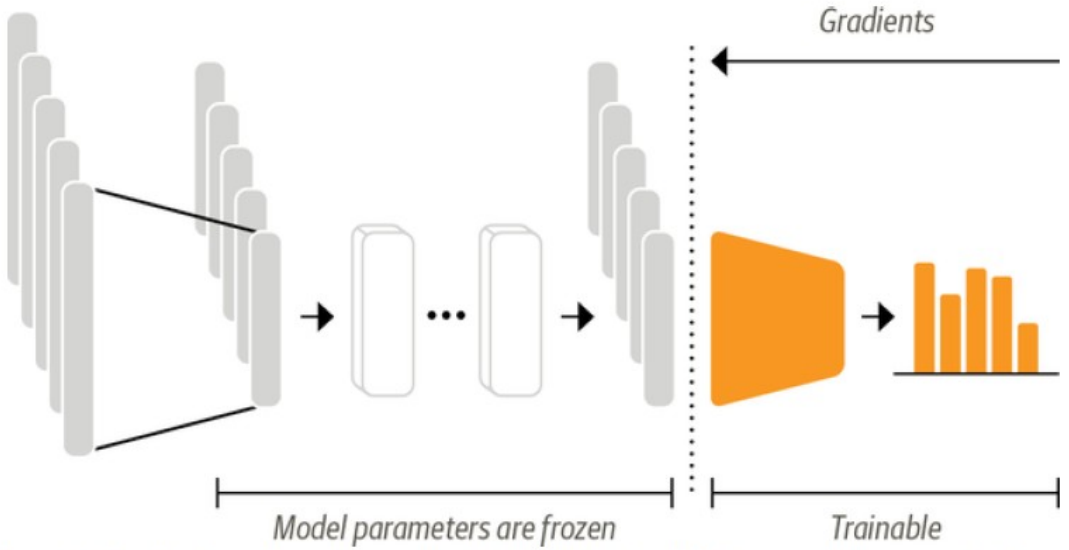


Figure 2-5. In the feature-based approach, the DistilBERT model is frozen and just provides features for a classifier

FINE-TUNING

- O modelo transformer é “levemente” treinado para se adequar à nova tarefa.

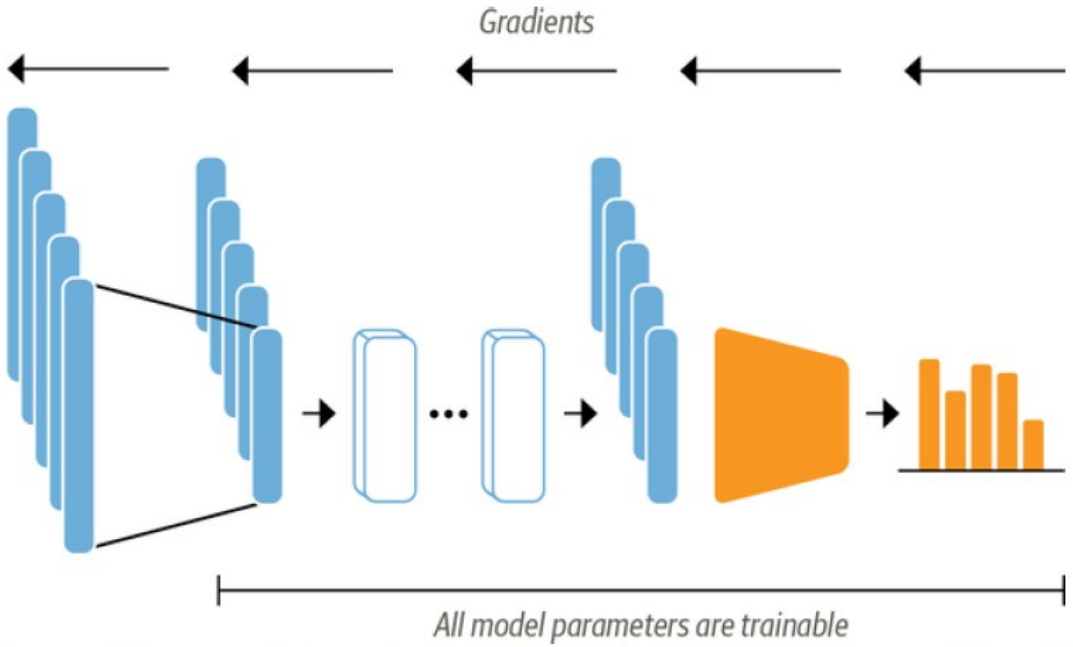
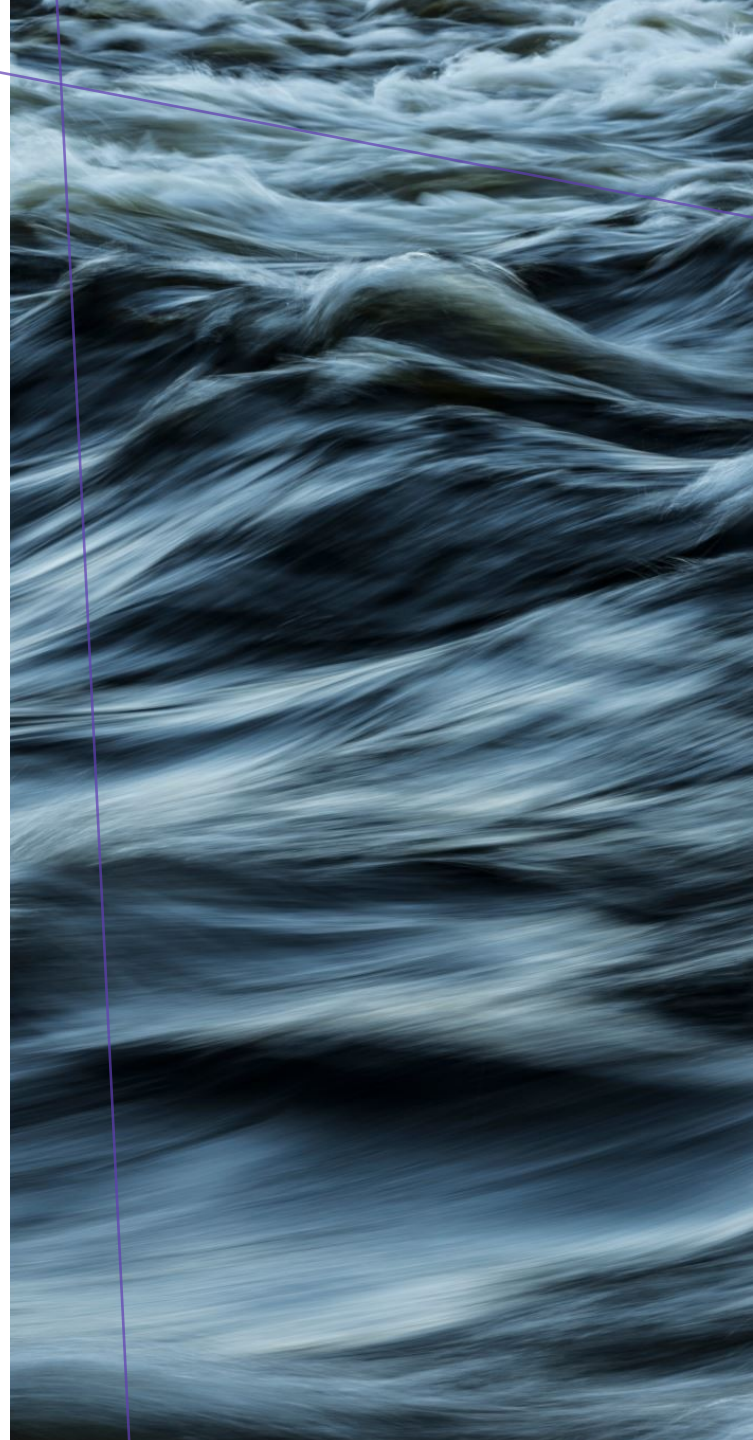


Figure 2-6. When using the fine-tuning approach the whole DistilBERT model is trained along with the classification head

HUGGING FACE



HUGGING FACE

- Plataforma de ferramentas de IA.
- Alimentada pela própria comunidade de usuários.
- Particularmente importante para o uso de modelos **transformers.**
- Disponibiliza uma série de tecnologias de PLN
  - Modelos transformers
  - Tokenizers
  - Conjuntos de dados
  - Métricas
  - Pipelines
- Permite o fácil compartilhamento dessas tecnologias e resultados.

AUTOCLASSES

- Uma AutoClasse automaticamente carrega a arquitetura correta e o checkpoint de um determinado modelo.
- Principais AutoClasses:
  1. AutoTokenizer
  2. AutoModel: arquiteturas-padrão para tarefas de PLN (AutoModelForQuestionAnswering, AutoModelForSequenceClassification, etc.)
  3. AutoFeatureExtractor
  4. AutoConfig
- https://huggingface.co/docs

REFERÊNCIAS

- Vaswani, Ashish, et al. “Attention is all you need.” *Advances in neural information processing systems 30* (2017). https://arxiv.org/pdf/1706.03762.pdf
- Devlin, Jacob, et al. "Bert: Pre-training of de transformers for language understanding." arXiv:1810.04805 (2018). https://arxiv.org/pdf/1810.04805.pdf
- Explicação detalhada dos transformers: https://jalammar.github.io/illustrated-transformer/
- Livro que usei como referência:

- paulopirozelli@gmail.com